ASTREA Framework: Development of Adaptive Monitoring Systems for Dynamic and Mobile Environments

Sara Balderas-Díaz

Doctoral Thesis

PhD Programme in Information and Communication Technologies

Supervisors
Dr José Luis Garrido
Dr Gabriel Guerrero-Contreras

UNIVERSIDAD DE GRANADA

ASTREA Framework: Development of Adaptive Monitoring Systems for Dynamic and Mobile Environments

UNIVERSIDAD DE GRANADA

Sara Balderas-Díaz

Supervisors: Dr José Luis Garrido
Dr Gabriel Guerrero-Contreras

Department of Software Engineering
University of Granada

PhD Program in Information and Communication Technologies

January 2022

Editor: Universidad de Granada. Tesis Doctorales
Autor: Sara Balderas Díaz
ISBN: 978-84-1117-245-5
URI: http://hdl.handle.net/10481/73158

«Science and everyday life cannot and should not be separated. Science, for me, gives a partial explanation of life. In so far as it goes, it is based on fact, experience and experiment.»

Rosalind Franklin

Resumen

Avances en las tecnologías ofrecen nuevas oportunidades en múltiples ámbitos. La aceptación y predisposición por el uso de las mismas, por parte de los *stakeholders*, también es un factor clave. Estas circunstancias han impulsado la evolución y desarrollo de soluciones o alternativas complementarias en los ámbitos de la salud (eSalud), vida cotidiana asistida por el entorno (Ambient Assisted Living, AAL) e internet de las cosas (Internet of Things, IoT), en los que el uso de sensores no intrusivos y *wearables* (como dispositivos ergonómicos con sensores incorporados) se ha convertido en la piedra angular de la recopilación de datos.

En los últimos años, se ha desarrollado una plétora de sistemas de monitorización de propósito específico para el control de enfermedades o situaciones concretas. Cabe mencionar que existe una ausencia de estándares o *frameworks* que faciliten su diseño y desarrollo. Sin embargo, si bien es cierto que tales sistemas comparten características, funcionalidades, objetivos, así como tecnologías específicas. Tampoco existen soluciones concretas y validadas que soporten el despliegue automatizado de un sistema de monitorización en una infraestructura de red móvil y dinámica. En relación con esto, cabe destacar que los cambios en los sistemas de monitorización son frecuentes por varias razones, tales como que las necesidades de los usuarios que están siendo monitorizados cambian, o que se requiera incorporar sensores/*wearables* así como funcionalidad al sistema del mismo modo que podría ser modificada o sustituida. Queda implícito, que estos cambios suponen el tener que volver a desplegar el sistema, a ser posible, en tiempo de ejecución para que las adaptaciones o actualizaciones se implanten de la forma más inmediata.

Algunas de estas circunstancias, pueden ser paliadas pues ciertos cambios se pueden prever y se pueden introducir mecanismos de adaptación y autoadaptación para ofrecer soluciones, en principio ad-hoc, pero que podrían favorecer la extensibilidad de los sistemas de monitorización. No obstante, es habitual, el tener que posponer estos cambios a versiones posteriores de los sistemas.

En lo que respecta a la recopilación de datos, a pesar de ser objeto de estudio durante años, aún existen desafíos principalmente derivados por la inestabilidad de las conexiones en entornos móviles, la congestión de la red, y las limitaciones en la capacidad de almacenamiento

y de energía de los dispositivos móviles. En este trabajo, con entornos móviles, nos referimos a redes móviles y dinámicas. Por una parte, con móvil nos referimos a elementos (o dispositivos) móviles que forman parte de la infraestructura de la red y que también pueden tener la capacidad de desplazarse por un área. Por otra parte, el término dinámico comprende, que se puedan incorporar nuevos elementos (o dispositivos) y que estos formen parte de la infraestructura de la red pero también el que las conexiones entre ellos puedan cambiar, incluso el que puedan ocurrir desconexiones. Ambas circunstancias provocan cambios en la infraestructura de red.

En esta tesis se propone ASTREA, un *framework* como solución integral de tres pilares ya introducidos y que pueden resumirse como sigue: 1) el diseño y desarrollo de sistemas de monitorización que incluyen sensores y *wearables*; 2) el despliegue automatizado del sistema en una red móvil y dinámica en tiempo de ejecución, así como su reconfiguración posterior al despliegue, lo que implica adaptaciones o actualizaciones en los sistemas, enfatizando la importancia de mantener la escalabilidad localizada; y 3) la recolección de datos incluyendo elementos móviles como portadores y sin incluirlos.

ASTREA ha sido creado como resultado del estudio previo de varios dominios de aplicación, del diseño y desarrollo de sistemas de monitorización en colaboración con expertos (psicólogos, médicos, enfermeros, matronas, educadores y terapeutas) y que han sido aplicados en entornos concretos. Esta investigación y desarrollos previos han favorecido la identificación de aspectos y características comunes a nivel funcional, de dispositivos, y de requisitos de calidad que se deben garantizar para el correcto funcionamiento de tales sistemas. Concretamente, se han desarrollado sistemas de monitorización aplicados directamente en entornos reales de pacientes que sufren Lupus Eritematoso Sistémico (LES) y Fibromialgia (F), y mujeres embarazadas (y sus parejas) con fetos Pequeños para la Edad Gestacional (PEG) o bebés nacidos con bajo peso.

Los sistemas de monitorización nos han permitido la recopilación de datos fisiológicos y datos del contexto (factores ambientales) mediante sensores y *wearables*. Estos datos obtenidos de forma objetiva han sido analizados de forma conjunta con otra información obtenida de forma subjetiva, mediante cuestionarios estandarizados o definidos por expertos (médicos, psicólogos/as, enfermeros/as, y matronas). Los sistemas desarrollados tienen un diseño y arquitectura que favorece el que puedan extenderse con el objetivo de que se puedan reutilizar en otros dominios de aplicación específicos.

La arquitectura de ASTREA es distribuida, se basa en servicios y microservicios, y soporta dos mecanismos implementados: (1) despliegue, y la propagación de adaptaciones y actualizaciones para extender los sistemas de monitorización de forma autónoma y en tiempo de ejecución; y (2) la recopilación de datos. Ambos mecanismos están operativos en una infraestructura de red inalámbrica y móvil. También incluye un repositorio de servicios que

almacena los microservicios que pueden ser desplegados en los dispositivos que forman parte de la red, de acuerdo a sus características. En ASTREA se pueden utilizar sensores o *wearables*, para la recogida de datos, pero también existen nodos, siendo estos últimos una agrupación de los dispositivos previos, que además dispone de un dispositivo con unas capacidades computacionales mínimas.

ASTREA gestiona los sistemas de monitorización como casos e incluye un editor visual para que determinados usuarios conocedores del dominio, puedan diseñar y desarrollar los casos de forma más rápida y sencilla, abstrayéndolos de cuestiones técnicas de más bajo nivel. El editor visual cuenta con una paleta en la que pueden seleccionar, arrastrar y soltar en un lienzo central diferentes elementos tales como los dispositivos (sensores/*wearables*) que va a incorporar el sistema u operaciones a partir de las cuales se define la funcionalidad que integrará, en base a los microservicios alojados en el repositorio y que, junto a lo anterior, favorece el reúso. Además, para ilustrar las capacidades y viabilidad del *framework* ASTREA, se implementa y se despliega en un entorno de pruebas real.

Finalmente, en la evaluación de ASTREA se ha utilizado el simulador ns-3 y tecnologías BonnMotion. Se analizan diferentes aspectos de la transmisión de datos y tiempos de propagación que supondría el envío de la especificación de un caso y los microservicios asociados para la composición del sistema de monitorización, en una red con dispositivos heterogéneos, dinámica, inalámbrica y móvil. En la misma red, se han evaluado las tasas de éxito en la recopilación de datos considerando la transmisión de datos en crudo versus la transmisión de información útil (datos procesados), la priorización de los datos frente a la no priorización, así como el impacto en el consumo de energía de los dispositivos.

Abstract

Advances in technology offer new opportunities in multiple fields. The acceptance and predisposition to use them by stakeholders is also a key factor. These circumstances have driven the evolution and development of complementary solutions or alternatives in the fields of eHealth, Ambient Assisted Living (AAL) and the Internet of Things (IoT), where the use of non-intrusive sensors or wearables (as ergonomic devices with built-in sensors) has become the cornerstone of data gathering.

In recent years, a plethora of specific purpose monitoring systems has been developed for the control of certain diseases or situations. It is worth mentioning that there is a lack of standards or frameworks to facilitate their design and development. There are also no concrete and validated solutions that support the automated deployment of a monitoring system in a mobile and dynamic network infrastructure. Concerning this, it is worth noting that changes in monitoring systems are frequent for various reasons, such as the needs of the users being monitored change, or sensors/wearables as well as functionality needs to be added to the system in a way that could be modified or replaced. Implicit in these changes is that the system will have to be redeployed, if possible at runtime so that the adaptations or upgrades can be implemented as quickly as possible.

Some of these circumstances can be mitigated because certain changes can be envisaged and adaptation and self-adaptation mechanisms can be introduced to offer solutions, in principle ad-hoc, which could favour the extensibility of monitoring systems. However, it is common to have to postpone these changes to later versions of the systems.

As far as data gathering is concerned, despite being the subject of years of study, challenges still exist mainly due to the instability of connections in mobile environments, network congestion, and limitations in the storage and power capacity of mobile devices. In this work, with mobile environments, we refer to mobile and dynamic networks. On the one hand, by mobile, we mean mobile elements (or devices) that are part of the network infrastructure and that may also have the ability to move around an area. On the other hand, the term dynamic comprises the fact that new elements (or devices) can be added and become part of the network

infrastructure, but also that the connections between them can change and disconnections can occur at any time. Both circumstances lead to changes in the network infrastructure.

This thesis proposes ASTREA, a framework as an integral solution of three pillars already introduced and which can be summarised as follows: 1) the design and development of monitoring systems which include sensors and wearables; 2) the automated deployment of the system in a mobile and dynamic network at runtime, as well as its post-deployment reconfiguration, which implies adaptations or upgrades to systems, emphasising the importance of maintaining localised scalability; and 3) the data gathering which comprises mobile elements as carriers and without including them.

ASTREA has been created as a result of the prior study of various application domains, the design and development of monitoring systems in collaboration with experts (psychologists, doctors, nurses, midwives, educators and therapists) and which have been applied in specific environments. This research and previous developments have favoured the identification of common aspects and characteristics at the functional level, device and quality requirements that must be guaranteed for the correct functioning of such systems. Specifically, monitoring systems have been developed for direct application in real-life scenarios of patients suffering from Systemic Lupus Erythematosus (SLE) and Fibromyalgia (F), and pregnant women (and their partners) with Small-for-Gestational-Age (SGA) foetuses or low birth weight babies.

Monitoring systems have enabled us to collect physiological data and contextual data (environmental factors) through sensors and wearables. These objectively obtained data have been analysed together with other subjectively obtained information through standardised or expert-defined questionnaires (doctors, psychologists, nurses and midwives). The systems developed have a design and architecture that favours extensibility with the objective of reusability in other specific application domains.

ASTREA's architecture is distributed, based on services and microservices, and supports two mechanisms implemented: (1) deployment, and propagation of adaptations and upgrades to extend the monitoring systems autonomously and at runtime. Both mechanisms are operational in a wireless and mobile network infrastructure. It also includes a service repository that stores the microservices that can be deployed within the devices part of the network, according to its capabilities. In ASTREA, sensors or wearables can be used to collect data, but there are also nodes, the latter being a group of the previous devices, that also have a device with minimum computational capabilities.

ASTREA manages the monitoring systems as cases and includes a visual editor so that certain users who are familiar with the domain can design and develop the cases in a faster and easier way, abstracting them from lower-level technical issues. The visual editor has a palette

from which different elements can be selected, dragged and dropped onto a central canvas, such as the devices (sensors/wearables) that the system will incorporate or operations from which the functionality to be integrated is defined, based on the microservices hosted in the repository and which, together with the above, favours reuse. Furthermore, to animate the capabilities of ASTREA framework, it is implemented and deployed in a real test environment.

Finally, in the evaluation of ASTREA, the ns-3 simulator and BonnMotion technologies. Different aspects of data transmission and propagation times required for sending a case specification and the associated microservices for the composition of the monitoring system, in a network with heterogeneous, dynamic, wireless and mobile devices, are analysed. In the same network, data gathering success rates have been evaluated considering the transmission of raw data versus the transmission of useful information (processed data), data prioritisation versus non-prioritisation, as well as the impact on the energy consumption of the devices.

Table of contents

List of figures

List of tables

Part I

Introduction, Foundations and Related Work

Chapter 1

Introduction

Chapter Abstract

Monitoring systems have become a common tool in many application domains (e.g., health, home, military, climate action, environment and resource efficiency). Most of them share requirements, objectives and technology, but they also have specific particularities. Advances in sensors and wearable devices collect data "everywhere" and "anytime". The Wireless Sensor Networks (WSNs) have driven the development of more complex networks that include mobile elements (i.e., portable devices, sensors or wearables). Cloud paradigms (e.g., edge computing, mobile cloud computing) have gained traction in recent years but it is important to stress the importance of localised scalability. Maintaining localised scalability is a reference point for the main objective of this thesis which is to provide an integral solution to common problems (i.e., speed up prototyping, post-deployment modifications and data gathering) of generic monitoring systems.

This chapter comprises an overview of the ideas, trends, concepts and considerations that have been studied in more depth during the development of the thesis. It also describes the motivation, objectives proposed and methodology followed to achieve them as well as the structure of the manuscript.

Chapter Contents

1.1 Introduction

The socio-economic changes and advances in technology are opening new opportunities in the field of health (eHealth) [3], Ambient Assisting Living (AAL) [58] and Internet of Things (IoT) [100]. A large number of devices support the access to the information "everywhere" and "anytime" which promote the development of alternatives based on software systems alleviating deficiencies and providing satisfactory solutions [181].

Modern eHealth, AAL and IoT systems are increasing in complexity due to internal and external factors, such as contextual conditions, user needs, modifications required in the system post-deployment, and services and resources availability. Therefore, this plethora of issues causes difficult-to-predict situations at design time before system deployment but these kind of systems also have similarities in common [258, 114]. Concerning to these factors, it is worth noting that the emphasis on building intelligent environments is driving the monitoring of context-specific information [70]; that the rise of the adaptation concept highlights the importance of making modifications at run-time and that it is feasible to [218, 211]; and that to gather the data, it is required a software infrastructure supporting the data management, sensors and actuators, and also the heterogeneity of the devices [83].

From a software point of view, the Service Oriented Architecture (SOA) has been claimed as the most suitable architectural approach for IoT, and more recently Microservices Architecture (MSA) is exhibiting a great potential [52, 50]. Both SOA and MSA support the decomposition of a system into services, although several concerns such as deployment, user interface, flexibility, management, scalability and service size present certain differences [51]. SOA was initially designed to operate in static environments but not straightforwardly in dynamic ones like IoT, in part because SOA relies on costly resource-consuming protocols [41]. On this basis, the combination of SOA with specific characteristics of the field of Autonomous Computing [259], resulting in adaptive or selfadaptive architectures, has been gaining importance in the research community because of the need to support autonomy in all phases of the service life cycle [1]. In addition, MSA is exhibiting a great potential in IoT [52, 50].

Addressing this plethora of issues can be complex and there are many challenges but also numerous approaches in research that addressed them individually [270]. It is also well known that End-User Development (EUD) helps to provide abstraction and tools to adapt the complexity in the development of domain-specific systems to end-users or domain-specific developers, who know best the needs and the specific requirements of applications in their domain [27]. It is worthy to mention that visual editing approaches increase the abstraction level and thereby, users can focus only on the functional requirements of their systems in

an intuitive way, rather than on the specifics of the underlying technology (communication protocols, programming languages or development APIs) [119].

From a network perspective, Mobile Wireless Sensor Networks (MWSNs) include mobile nodes (or elements) that can help to increase coverage range, storage capacity, connectivity, network reliability, lifetime, data collection, data gathering, and to reduce power consumption [269]. The term mobile will henceforth be considered to refer to a portable device, sensor or wearable (i.e., an element with built-in sensors) that may or may not being moved, while the term node will encompass the possibility of being an element or a grouping of them. In addition, these mobile elements (MEs) could help to perform in-network preprocessing operations in order to optimise the Wireless Sensor Network (WSN) performance mainly to get meaningful information leveraging localised scalability and edge computing paradigm [206, 219]. The MEs could be the intermediaries between data sources and cloud data centers [219].

In conclusion, the ensemble of all the pieces (i.e., components, services, microservices, tools, mechanisms, techniques, methodologies, etc.) could be framed in a framework that would promote reusability, facilitate the solution to related problems and development tasks, and where a set of components would be offered to be utilised by the users who would not need to know how they are implemented [121, 122]. Therefore, experts can encapsulate their knowledge in a framework and even standardise approaches with methodological frameworks which provide a guide, a guideline, or a series of steps to carry out a process [152, 94].

1.2 Motivation

The motivation for this research is to respond to the needs that arise in modern computing environments. The following describes the details that have inspired and encouraged us to combine relevant technologies and approaches to provide a comprehensive solution.

In health and AAL domains, associations, institutions and medical staff agree that Information and Communication Technologies (ICTs) help to provide a more efficient, effective, reliable and faster service, health care, diagnosis, treatment and follow-up, reduce waiting times, saves costs, and citizens obtain useful information and alternative diagnoses [89, 107]. The World Health Organization (WHO) Global Observatory for eHealth (GOe) defines that mobile devices in combination with wireless technologies had great potential to support health practices, and this is what is known as mobile health (mHealth) [199]. Currently, more than 2.5 billion people own their a mobile device and Bring your Own Device (BYOD) trend are being well accepted by the population [195, 136].

In recent years, the use of sensors and wearables is growing fast, and their low-cost combination with communication technologies is making easier the deployment of pervasive monitoring systems, the development of smart homes and environments that make remote patient monitoring feasible. These systems are especially useful for patients with limited access to hospitals or who require routine check-ups, or elderly people [24, 192]. The AAL comprises many ideas but one of the most important is Ambient Intelligence (AmI) [49]. AmI promotes unconscious human interaction with the environment (or with its surroundings) which is related with to ubiquitous computing paradigm [257]. AAL involves a wide variety of sensors technologies, services and systems to improve the quality of life of people and their relatives [49].

In the meantime, IoT has become one of the most powerful technologies for interconnecting numerous heterogeneous smart objects (e.g., devices, sensors, actuators, smartphones, cameras, smart wheelchairs or robots) creating large and dynamic networks. The number of devices connected to the Internet increases every year and is expected to reach 75.44 billion by 2025 [5, 205]. In fact, IoT is considered a driving force for the development of smart cities and the support of specific application domains (e.g., healthcare, environmental monitoring, vehicles, transportation, smart homes, tourism, market, smart retail, agriculture, industry or energy) [5, 205]. IoT also give rise to Internet of Medical Things (IoMT) [222, 193].

Wireless Sensor Networks (WSNs) are perceived as a feasible solution to large-scale tracking and monitoring in many areas of the domains already introduced [144, 253, 269]. Traditional WSNs were composed of a set of static devices, with pre-determined positions, which can connect by single-hop to cover partial areas, or multi-hop paths to cover large areas [265]. As a step forward, MWSNs emerged [269]. WSNs also comprise Body Sensor Networks (BSNs) that monitor corporal human conditions (i.e., ageing population, dependent people, chronic patients and vulnerable groups) [93]. As a result, wireless connections have become a backbone of application and systems, and they have been ranked among the top 10 technologies that are changing the world [140, 105].

Related to individual issues, context and quality information can be gathered by devices, sensors and wearables, and even processed by themselves [46]. It is even easier to modify the requirements or behaviour of the monitoring systems at run-time, and address the dynamic modifications that can occur as result of considering the context information [114].

Several architectural solutions have been proposed to improve the availability of the services deployed and minimising battery consumption in systems through service replication and selfconfiguration techniques [96, 97]; supporting a large number of users and heterogeneous devices, collaborating and sharing information (from smart cars to sensors or wearables) [249];

enhancing interoperability, flexibility, extensibility, maintenance and low coupling through services and microservices [230]; modelling the opportunistic properties of the IoT systems, which remains evolving [47]. Data gathering approaches have also been proposed. A distinction must be made between data collection and data gathering. The term data collection is more commonly used when referring to the taking of measurements through a device while data gathering term is closer to data gathered from sensors that can be transported by MEs or send directly to a centralised unit for intensive processing. When data is being gathered and the volume of data is large, transmission may be impractical, bottlenecks can be caused, and certain systems may have latency requirements that can not be guaranteed due to changes in the network [55, 253]. In these circumstances, applying an edge computing approach could help to alleviate these problems. In this way, data fusion combines data from multiple sensors, speeds up the transmission process, and favours the collection of meaningful, reliable, accurate, complete information thus maintaining the relevance of the concept of localised scalability, allowing customisation of applications [253, 24]. It should be noted that the energy required to do computation is 1000 times less than that needed to transmit [253]. Therefore, edge devices are sometimes used to alleviate both issues due to one of their main purposes is bring data computation closer to source nodes [219]. Therefore, source nodes perform first-order operations (e.g., sensing, filtering, aggregation, etc.) on raw data and can be connected to a fog node which frequently performs more complex operations [219, 55]. WSNs can include a large number of sensors and even various base stations, and monitoring systems generally share similar operations, features, and several tasks can be reused within different applications [137].

However, it should be noted that systems must be able to operate in highly dynamic networks with unpredictable network topology changes and unreliable communication channels, but there are numerous challenges to be solved and static environments (or networks) are still common [47]. Solutions for representation, architecture, self-* methods to discover, operate, manage, compose, coordinate and deploy the services, microservices or smart objects at runtime are still needed [1, 50] and quality of attributes (i.e., inter alia, availability, reliability, performance and scalability) must be taken into account for efficient service provision [72]. Furthermore, society demands that distributed systems are available 24/7, require no human supervision and minimal maintenance effort [201]. These make it essential to look for versatile, flexible, resilient and robust design and implementation solutions [201].

1.3 Objectives

The main objective of this PhD thesis is to provide solutions that facilitate the development of monitoring systems, their deployment, adaptations and upgrades in dynamic and mobile environments, and performance improvements in data centralisation and indirectly in energy consumption. To achieve this, the following goals are proposed:

- **Goal 1:** To analyse the characteristics, hardware elements, infrastructure and software architectures that are incorporated in monitoring and control systems for specific application domains.

  Several application domains will be studied more in deep to check whether they truly share commonalities or characteristics.

- **Goal 2:** To study software approaches and architectures that are suited to dynamic environments (SOA, SOA 2.0, EDA, ROA, agent architecture, etc.), frameworks and methodologies that favour software reuse and development, as well as cloud model paradigms.

  From this objective we hope to gain knowledge that will help us in the development of our proposal.

- **Goal 3:** Literature review. The search should be focused on proposals for both ad-hoc and generic monitoring systems designs. In addition, proposals for solutions to specific challenges in mobile and dynamic networks, and mechanisms to support system modifications or adaptations should be explored.

- **Goal 4:** To design and implement an architecture that supports the deployment of a monitoring system, with minimum human intervention (i.e. autonomously) in a dynamic network with and without mobile elements. It also supports upgrading the system once deployed, at runtime. The upgrading of the system would allow to introduce modifications or adaptations by adding, modifying or removing functionality and/or sensors/wearables, offering continuous and personalised monitoring to users according to their current conditions and state. Furthermore, the architecture should support in-network preprocessing and centralisation of data. In addition, note that systems developed using this architecture should support the inclusion of self-adaptation mechanisms.

- **Goal 5:** To study the supporting infrastructures (sensors and wearables), communication and integration technologies, simulators for deployment.

Communications and technologies evolve, just as simulator versions change. With this goal in mind, the aim is to study the different options and acquire the skills to select the most appropriate ones to incorporate in our research.

- **Goal 6:** To determine and provide solutions to manage monitoring systems. Consideration should be given to the gathering of context information (i.e., physiological and environmental context information) to meet the objectives of user monitoring. Moreover, it could be of interest to consider devices state in order to satisfy the quality of service (QoS).

- **Goal 7:** To propose a framework, taking into account which has been studied in the goals described previously, that maximises and abstracts reuse issues in order to facilitate the design and favour the rapid development of monitoring systems. Furthermore, it should support both modifications within monitoring systems and data gathering actions. For the data gathering, among others, it should be considered the inclusion of in-network preprocessing actions in order to optimise WSN performance in terms of increasing success rate of meaningful information, and minimising bandwidth and energy consumption.

- **Goal 8:** To show the behaviour of the proposal which will be illustrated by a case study. In addition, to test the validity of the proposal in dynamic and mobile environments.

1.4 Methodology

First of all, we will perform a study of different areas that frames each of the proposed objectives. This will encourage the development of new approaches that complement existing researches.

Experts from the health sector will be involved. They will help to define the needs and constraints of each specific case study. We will analyse the problem, define the requirements, propose a design and develop the proposal. Subsequently, we will proceed with the evaluation of the proposal in real-life scenarios, including standardised evaluation metrics. In order to achieve this, we will consider four of the six common phases of software technologies [216]:

- *"Basic research"* consists on create the structure of the problem from research concepts and ideas.

- *"Concept formulation"* through informal communication of ideas between the researchers involved and proposing solutions to concrete sub-problems.

- *"Development and extension"* generalising the problem.
- *"Internal enhancement and exploration"* extending the approach to other domains, using in real situations, developing materials, and showing results.

Dooley [69] shows that a case study can hold multiple cases, while considering both quantitative and qualitative data and various research paradigms. He also argues that they help developers in understanding more complex issues and allow different methodologies to be combined in data collection, data comparison and validation. Therefore, we will take such reflections into consideration in addition to those of Tichy's reflections [238]. From Tichy's reflections it is worth highlight his concern about the speed of change in technology. We will therefore try to follow his recommendation, anticipating changes that may be required, in order to provide solutions that are sustainable or that can accommodate the implementation of modifications.

We will follow the Software or Systems Development Life-Cycles (SDLC) [198] which *"is a conceptual framework or process"* that considers the different stages of the development of an application that can be summarised as initial study, deployment and maintenance. Specifically, we will select the waterfall model but if necessary, we will combine, substitute or adapt it with other existing models in the literature. The waterfall model comprises the *"evaluation, requirements, analysis, design, development, validation, and development"* stage, and it is bi-directional, i.e. it allows moving from one stage to the next but also back to the previous one or even to its predecessor.

The framework proposal will follow the same methodology but with the experience and knowledge gained from the previous cases.

The proposals and results obtained from the research will be submitted for review, through participation in different scientific dissemination media, such as forums, national and international conferences, and journals of recognised prestige. Shaw has already advanced that an effort is required to validate the work, both at the individual level of research and in the long term (e.g., in projects) [216].

1.5 Manuscript Structure

Next, the structure of the PhD thesis is described.

- In Chapter 1, some ideas and relevant concepts are introduced, and the motivation points, the main objective and the goals set to achieve it are exposed. It also describes the methodology, the structure of the document.

- Chapter 2 includes a review of the foundations and technologies considered as a reference and/or relevant to achieve the proposed goals.

- Chapter 3 presents a selection of relevant proposals related to the main objective to achieved during the development of this thesis.

- In Chapter 4, we introduce the application domains addressed, as well as the systems developed and used by psychologies and the medical teams of Hospital Universitario Virgen de las Nieves (de Granada), Hospital Universitario Virgen del Rocío (de Sevilla), and researchers from University of Granada. It also summarises some of the results obtained from the derived studies in which we have been directly involved. The study of these domains, and the system proposals provided, have served as a basis for identifying common functionalities shared by the plethora of monitoring systems designed ad-hoc. As result of the study of several application domains, as well as the development of several monitoring systems for some of them, the foundations of the ASTREA framework (described in Chapter 5) are laid.

- In Chapter 5, we present ASTREA framework as main contribution. ASTREA aims to be an integral solution to speed up prototyping of monitoring systems, and supporting post-deployment adaptations and upgrades of these monitoring systems at runtime as well as data gathering. These monitoring systems will be deployed in mobile and dynamic environments. Useful aspects of design processes, hardware devices and functionalities have been embedded within ASTREA framework. ASTREA includes a visual editing tool for designing the monitoring systems by domain-specific software developers who can focus on system design. It also comprises a mechanism for the autonomous deployment, and propagation of adaptations and upgrades with minimal human intervention of the systems which are already monitoring. Furthermore, ASTREA addresses how to increment data gathering success rate based on in-network preprocessing by data gathering mechanism. Moreover, it also includes the option to prioritise the data that prevails under adverse conditions (e.g., no storage available). This chapter provides detailed information about ASTREA, presents a case study, and includes implementation and deployment in a real test environment.

- Chapter 6 presents the material and methods of the ASTREA evaluation for data transmission and propagation time. To achieve it, two studies are described, results obtained are shown and discussed.

- Chapter 7 highlights the contributions of the thesis, summarises the objectives achieved and outlines the future work.

1.6 Publications

The publications obtained as result of this research work are shown below.

Journals

- **Using Actigraphy and mHealth Systems for an Objective Analysis of Sleep Quality on Systemic Lupus Erythematosus Patients**. S. Balderas-Díaz, M. P. Martínez, G. Guerrero-Contreras, E. Miró, K. Benghazi, A. I. Sánchez, J. L. Garrido, G. Prados. *Methods of Information in Medicine* 2017 vol: 56 (2) pp: 171-179. [JCR Q3; IF 1.531] 10.3414/ME16-02-0011

- **A Context-Aware Architecture Supporting Service Availability in Mobile Cloud Computing**. G. Guerrero-Contreras, J. L. Garrido, S. Balderas-Díaz, C. Rodríguez-Domínguez. *IEEE Transactions on Services Computing* 2017 vol: 10 (6) pp: 956 - 968. [JCR Q1; IF 4.418] 10.1109/TSC.2016.2540629

- **Self-adaptive deployment of services in mobile environments: a study of the communication reliability on the host election algorithm**. G. Guerrero-Contreras, S. Balderas-Díaz, C. Rodríguez-Domínguez, J. L. Garrido, A. Valenzuela. *Journal of Reliable Intelligent Environments* 2016 vol: 2 (4) pp: 197-207. 10.1007/s40860-016-0029-3

Book Chapters

- **Designing New Low-Cost Home-Oriented Systems for Monitoring and Diagnosis of Patients with Sleep Apnea-Hypopnea**. S. Balderas-Díaz, K. Benghazi, J. L. Garrido, G. Guerrero-Contreras, E. Miró. *ICTs for Improving Patients Rehabilitation Research Techniques* 2015. pp 210-221. 10.1007/978-3-662-48645-0_18

International Conferences

- **Design of an Adaptable mHealth System Supporting a Psycho-educational Program for Pregnant Women with SGA Foetuses**. S. Balderas-Díaz, M. J. Rodríguez-

Fórtiz, J. L. Garrido, M. Bellido-González, G. Guerrero-Contreras. *International Conference on Conceptual Modeling* 2021 pp: 125-135. 10.1007/978-3-030-88358-4_11

- **Integrating a Dual Method on a General Architecture to Self-Adaptive Monitoring Systems**. S. Balderas-Díaz, K. Benghazi, J. L. Garrido, G. P. M. O'Hare, G. Guerrero-Contreras. *Advances in Intelligent Systems and Computing* 2017 vol: 1 pp: 528-538. 10.1007/978-3-319-56535-4_54

- **Impact of Transmission Communication Protocol on a Self-adaptive Architecture for Dynamic Network Environments**. G. Guerrero-Contreras, J. L. Garrido, M. J. Rodríguez-Fórtiz, G. P. M. O'Hare, S. Balderas-Díaz. *Advances in Intelligent Systems and Computing* 2017 vol: 206 pp: 115-124. 10.1007/978-3-319-56538-5_12

- **Designing Configurable and Adaptive Systems in eHealth**. S. Balderas-Díaz, K. Benghazi, G. Prados, E. Miró *Proceedings of the 3rd 2015 Workshop on ICTs for improving Patients Rehabilitation Research Techniques* 2015 pp: 118-121. 10.1145/283-8-944-28389-7_3

- **An Approach Addressing Service Availability in Mobile Environments**. G. Guerrero-Contreras, S. Balderas-Díaz, C. Rodríguez-Domínguez, A. Valenzuela, J. L. Garrido. *Workshop Proceedings of the 11th International Conference on Intelligent Environments*, Prague, Czech Republic, July 15-17, 2015. vol: 19 pp: 46-57. 10.3233/978-1-61499-530-2-46

- **Dynamic Replication and Deployment of Services in Mobile Environments**. G. Guerrero-Contreras, C. Rodríguez-Domínguez, S. Balderas-Díaz, J. L. Garrido. *New Contributions in Information Systems and Technologies* 2015 vol: 353 pp: 855-864. 10.1007/978-3-319-16486-1_85

- **Self-adaptive Service Deployment in Context-Aware Systems**. G. Guerrero-Contreras, J. L. Garrido, C. Rodríguez-Domínguez, S. Balderas-Díaz. *Ubiquitous Computing and Ambient Intelligence. Personalisation and User Adapted Services*. UCAmI 2014. vol 8867, pp. 259–262. 10.1007/978-3-319-13102-3_42

- **Consistent Management of Context Information in Ubiquitous Systems**. G. Guerrero-Contreras, J.L., Garrido, S. Balderas-Díaz, C. Rodríguez-Domínguez. *Internet and Distributed Computing Systems - 7th International Conference*, (IDCS) 2014, Calabria, Italy, September 22-24, 2014. 10.1007/978-3-319-11692-1_16

- **Towards a Self-Adaptive Deployable Service Architecture for the Consistent Resource Management in Ubiquitous Environments**. G. Guerrero-Contreras, J. L. Garrido, K. Benghazi, S. Balderas-Díaz, C. Rodríguez-Domínguez. *Workshop Proceedings of the 10th International Conference on Intelligent Environments*, Shanghai, China, June 30 - July 1, 2014. vol: 18 pp: 206-217. 10.3233/978-1-61499-411-4-206

- **A service-based platform for monitoring and diagnosis of patients with SAHS symptoms**. S. Balderas-Díaz, K. Benghazi, J.L. Garrido, G. Guerrero-Contreras, E. Miró. *Proceedings of the 8th International Conference on Pervasive Computing Technologies for Healthcare*, PervasiveHealth 2014, Oldenburg, Germany, May 20-23, 2014. pp: 290-293. 10.4108/icst.pervasivehealth.2014.255365

Chapter 2

Foundations and Technologies

Chapter Abstract

Proposing a comprehensive solution to speed up the development of monitoring systems, supporting the ability to be modified in a dynamic network and mobile environment, and improving data gathering rates require to study the fundamentals related to each of the objectives of this thesis. This chapter provides a review of the most relevant aspects of these fundamentals. Adaptation and self-adaptation, autonomic computing and context-awareness have been key concepts for the development of user-centric systems and from which quality properties can be improved. Modularity, encapsulation and low coupling are software-level properties that must be preserved in order to enhance reuse, minimise dependencies between entities and facilitate modifications to monitoring systems. At this regard, services and microservices are highly recommended technologies in distributed environments and eHealth, AAL and IoT domains. Agent systems and multi-agent systems have also been reviewed. Software infrastructures (i.e., architectures, frameworks, agile practices, etc.) to host the functionality as well as networks infrastructures (i.e., sensors, wearables, mobile nodes, etc.) have been considered. Finally, we have review the evaluation tools (i.e., simulators) that allow us to assess the impact of monitoring systems configurations deployment and data gathering in dynamic and mobile networks.

Chapter Contents

2.1 Adaptation and Self-Adaptation

A priori, the design and development of software systems is dependent on the application domain and specific purpose. However, systems can be adaptive and/or self-adaptive [64]:

- *Adaptation* is defined as *"the ability of the application/system to modify and reconfigure as a result of (i.e. in reaction to) context changes [...] to deliver the same service in different ways when requested in different contexts and at different points in time"*; and
- *Self-adaptation* occurs when the application/system is autonomous or, in other words, it is able to act on its own carrying out the adaptation process without human intervention in response to changes in its context or operating environment. This approach has been proposed to address the increasing complexity management of systems after their initial deployment, adapting themselves in consideration of changes in their environment and user requirements leading to *Self-Adaptive Systems (SAS)* [260].

Here, it is worth mentioning the concept of autonomic computing [127] for system self-management achieving 24/7 system performance and maintenance according to the goals of an administrators but relieving it of the workload. Autonomic systems must continuously monitor themselves and update their components. To achieve it, this concept defines self-* to group four properties that will be carried out autonomously by the systems themselves:

- *Self-configuration* includes installation according to high-level policies, configuration, integration of complex, even error-prone systems, and monitoring of time consumed. It also includes automatic learning, taking into account the composition and configuration of the system when new components are included so that other components can use them or modify their own behaviour.
- *Self-optimisation* involves setting parameters improving system operations, efficiency or cost, active search for updates, and performance improvement.
- *Self-healing* focuses on detecting, diagnosing, and repairing localised problems resulting from bugs or failures, identified from log files analyses and supplementary data, and then installing patches and retest.
- *Self-protecting* defending from correlated problems and malicious attacks or cascading failures not corrected by self-healing as well as early detection of problems.

Salehie et al. [202] classify them as *major level* properties and group the following into a *primitive level* (Figure 2.1):

- *Self-awareness* refers to the fact that, since the system is continuously monitoring itself, it is aware of its internal state and its functioning [108].
- *Context-awareness* means that the system knows the environment in which it operates [208].

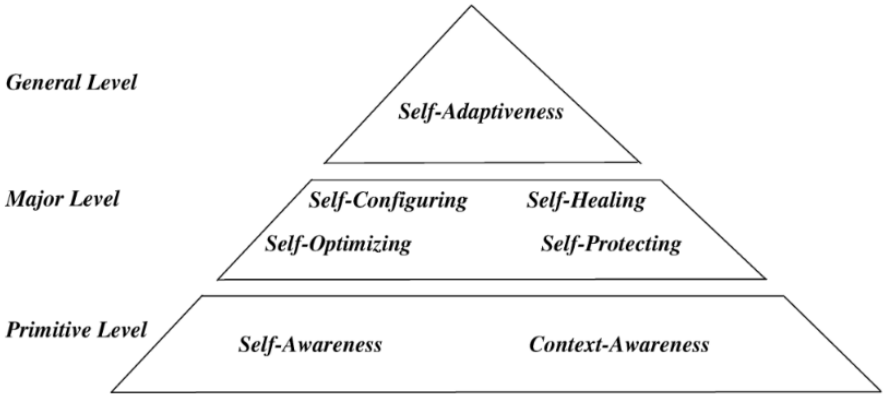


Fig. 2.1 *Hierarchy of the self-* properties proposed by Salehie et al. [202]*

2.1.1 Context

Anind K. Dey [65] proposed in 2001 the *context* definition as *"... any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves*. By taking into consideration this definition, Dey proposed the definition of *context-aware* software as *A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task*. In turn, the context information can be classified into different categories [263] as can be seen in Figure 2.2.

- *System context* is the information related with the system itself (e.g., CPU, network, status, etc.). This category collect information from the *Open System Interconnections (OSI) model: application*[263].
- *User context* is the information related with the user (e.g.,gender, age, weight, height, body mass index, medical history, personal state, etc.). It also can be considered the type of activity that the user is performing, the users involved in the activity and how to influence each other, the location of users, the place in which the user is, and the characteristics of their environment [139].

- *Environmental context* is the information related with the environment (e.g., temperature, luminosity, noise, weather, etc.) not included within the two previous (i.e., *system context* and *user context*).
- *Temporal context* is the information related with time (i.e., time, day, etc.).

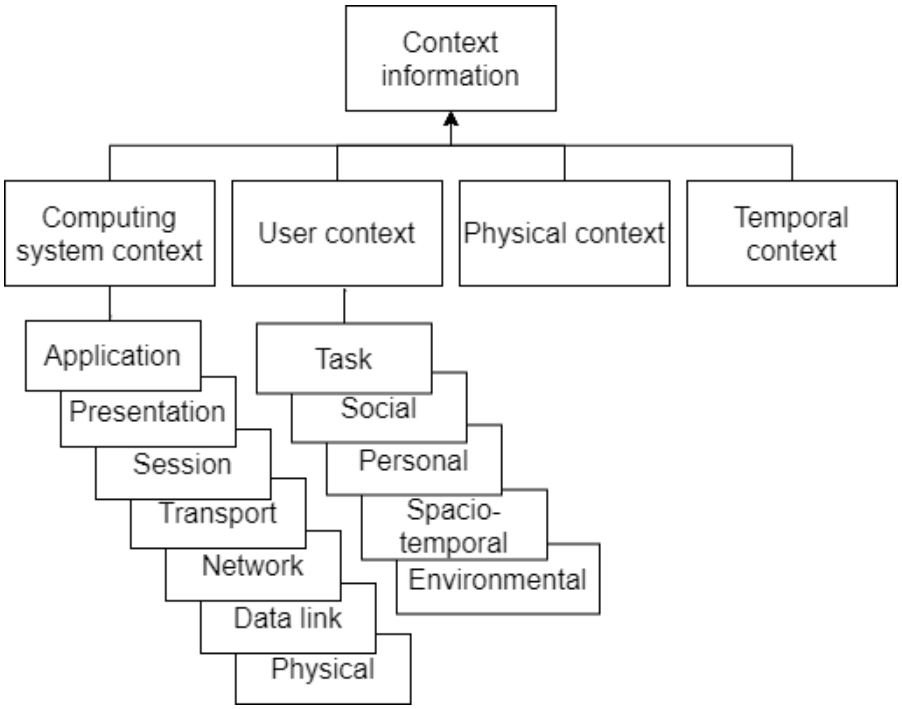


Fig. 2.2 *Classification of context information proposed by Wrona et al. [263]*

Adaptive systems are focused on specifying the full domain from early stages, which is more complex than the usual process but it provides clear advantages [64]. These systems must include monitoring information on their characteristics as well as the management of it, or in other words, the extraction and use of this information from the context or self-awareness to provide services to users [64], or to detect changes which could be relevant to improve the support provided at a point in time, and to improve quality properties [65, 178]. Several researchers even point out that from the early stages of design it is necessary to consider the infrastructure of devices (wearables, sensors, mobile devices, etc.) needed to collect information and satisfy system requirements [83, 252]. Depending on the origin of the information the system should manage and interpret it differently [139].

In recent years, a large number of works address self-adaptation considering context information, from different points of view, e.g. taxonomies, architectures implementing control

loops, adaptive requirements, etc. with the long-term goal of laying the foundations for the systematic development of future generations of self-adaptive systems [201].

2.1.2 Adaptation Dimensions

There are different perspectives and aspects to consider and which must be implemented to perform self-adaptation in SAS and adaption in general [135]. In Figure 2.3, Krupitze et al. present a taxonomy whose dimensions are related to questions to be asked during the implementation of the adaptation process. Specifically, the association of each of the questions and their dimension is as follows: *When?* related with *Time* dimension; *Why?* related with *Reason*; *Where?* related with *Level*; *What?* related with *Technique*; *Who?* related with *aimed at a type of automatic adaptation*; and *How?* related with *adaptation control*.

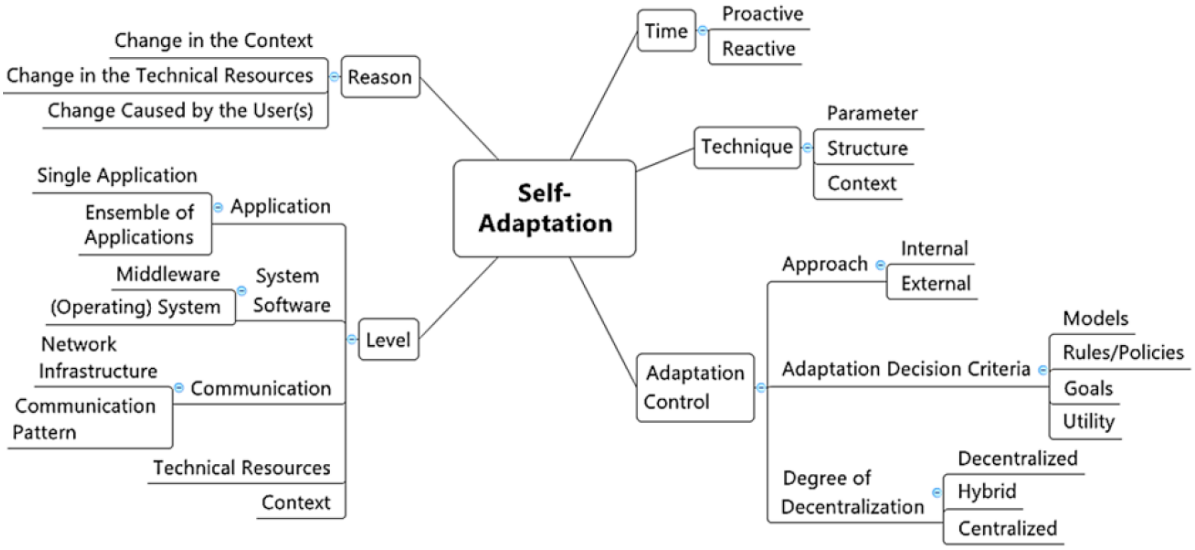


Fig. 2.3 *Taxonomy dimensions of self-adaptation proposed by Krupitzer et al. [135]*

The granular decomposition of perspectives included in each dimension of the proposal by Krupitzer et al. [135], combined and complemented with other proposals by other authors, is detailed below.

- *Time* dimension:
  - *Predictive* detects the need for adaptation before performance declines.
  - *Proactive*, an adaptation is carried out in order to improve performance without anticipating a drop in performance. This option is preferable by users.

- *Reason* dimension where an adaptation is a reaction to one or more changes. This also involve *knowledge* to consider information about the system, context and users, and several mechanisms (e.g., ontologies) are used according to the implementation that has been carried out.
  - *Context* such as changing a variable.
  - *Technical resources* such as changing a hardware component or hardware resources (e.g., a software fault, an alternative network connection or battery level of a device).
  - *User* or *users information* to know their short-term and long-term interests and preferences [102].
- *Level* at which to implement change, it could be any of the following:
  - *Application* correspond to a current instance that may be running on one device or distributed and running on multiple devices. It can also refer to the system. The adaptation could be to mute the phone if the person has a meeting that is detected by his or her calendar.
  - *System software* controllers the hardware. Applications are at the top of software systems. The adaptation could be to exchange software components at runtime.
  - *Communication* involves physical *network infrastructures* (e.g., routers) and *communication patterns* (e.g., events, publish/subscribe messages). Adaptation could consist of switching from WLAN to 5G.
  - *Technical resources* may cause, for example, a backup to be made.
  - *Context* may cause, for example, mute the microphones of the attendees at a meeting except for the speaker's microphone.
- *Technique* to specify *"What kind of change is needed?"* or *"What kind of action is needed?"* to perform the adaptation:
  - *Parameter* to adjusts specific parameters which may be simple but can be complex if the parameters are dependent on other parameters. Parameters can change at runtime and it could be defined policies to decide how and when the values change [145]. An example could be loading an image to a mobile device with lower quality due to bandwidth limitation. Kakousis et al. [125] indicate that this technique is also used to modify extra-functional properties (i.e., performance, reliability and other QoS properties), and Ketfi et al. [128] propose the following classification for QoS adaptation:

    - *Corrective adaptation* identifies the erroneous component and replaces it with a new version that provides the same functionality.
    - *Adaptive adaptation* to change the application to changes affecting its environment.
    - *Extending adaptation* with new components in response to new user functionalities not considered during development or at deployment.
    - *Perfective adaptation* is about improving and optimising the application components even when there are no errors.
  - *Structure* to exchange, reorganising, modify, add or remove components (or services, modules, etc.). McKinley et al. [151] call it *compositional adaptation* to refer to algorithms or system components that can be dynamically exchanged at runtime. In addition, *executable and compilable units* could be used to adapt or personalised the system, adding, removing or replacing units of code at deployment, load or runtime, such as modules (e.g., jars) [7]. *Code mobility* [125] consists of migrating or moving instances of programs, code or objects from one host to another at runtime. Mobility can be categorised as *strong* when code and execution state move and *weak* whether it only move code and some kind on initialisation is performed. Based on two models, *code pushing* (one unit sends code to another) or *code pulling* (retrieve and deploy a code), there are four paradigms supporting code mobility:
    - *Client/Server*: clients request the execution of a specific code to the server.
    - *Code On Demand (COD)*: clients can download code, an executable or an application, and then may even transmit it to other nodes. It is useful for upgrades and when it is not possible to pre-load full functionality due to limited device resources or unpredictable context changes.
    - *Remote evaluation* is used by nodes that push code to other nodes with higher computational capacity and delegate evaluation tasks (i.e., *resource demanding computations*) to them.
    - *Mobile agent* is an autonomous unit that is incorporated into a network to perform specific tasks. Its use is especially useful in networks where high availability is not guaranteed (i.e., disconnections).
  - *Context* refers to making changes to the context, e.g. modifying the context using actuators.

- *Adaptation control* to answer *"How to adapt?"* which involves monitoring systems and environment, analysis, planning and executing.

- *Approach* that distinguishes between *internal* and *external* approaches. With *internal* approach, the adaptation and system resources are linked whereas with the *external* approach they are not, which favours modulation and maintenance.
- *Adaptation decision criteria* includes methodologies and techniques mentioned in the previous dimension. The adaptation involves the implementation of changes that may affect system properties or variables without changing the structure or parts of the system, but adaptation may also affect the structure, interface, or components. Krupitzer et al. [135] distinguish between *models*, *rules* and *policies*, *goals*, and *utility functions*, and Kakousis et al. [125] granulate it on more levels such as *action-based adaptation*, *goal-based adaptation*, and *utility functions*, *case-based reasoning*, and *reinforcement learning*. Each of them can be described as follow:
  - *Models* describe the architecture of the running systems, and depending on the level of abstraction, adaptation or customisation can be generalised [10]. Here, at the highest level, Sabatucci et al. [201] introduces a metamodel that classifies smart systems into a specific category according to their properties. The metamodel reflects passive entities (e.g., software applications, physical devices, etc.) and environmental objects (e.g., physical or digital). Smart systems are known because *"owns a kind of smart entity"* to know their environment and act accordingly, and they may also posses an *Awareness Engine* to detect the system model at runtime and change the strategy built at design time supported by *Solution Builder* which works with a repository of predefined functionalities.
  - *Action-based (or rule-based) adaptation* is the most popular approach used for self-managing. Systems are defined as being in a state S at an instant t and if P (policy) is satisfied by an action a, it will occur a transition (probabilistic or deterministic) to another state $S2$. In addition, the concept of event-action rules [255] is used for dynamic reconfiguration of component-based architectures. Rule-based approaches has a binary decision (IF-THEN rules) limitation because the rule is evaluated as *true* or *false* which will be routinely checked considering the specification, and triggered autonomously when they are satisfied (i.e., the parameters are within the condition thresholds) in order to self-adapt the configuration of the architecture. However, Fuzzy Logic and fuzzy rules introduces multi-values, it means IF-THEN rules with simultaneously valid possibilities. This must be specified by developers at design time with the aim of trigger a particular adaptation mechanisms at runtime

as response to a context change. It is noteworthy that any possible context change that should trigger an adaptation must be associated with at least one adaptation rule. The more elements of the context we are considering, the more rules we need to specify. To conclude, *dynamic modification* and *evolution* of rules increases complexity and usually requires recompilation.

* *Goal-based adaptation* does not specify a transition from an state *S* to *S2*. It is up to the system to decide which adaptation causes the transition to the target state. This approach works with high-level goals closer to the human reasoning but requires the use of sophisticated planning and modelling algorithms to include this rational behaviour. Moreover, often is frequent the use of modules to monitor parameters and predictions. However, conflicting rules is not supported, nor is the comparison of mechanisms where more than one state can achieve the same goal. Finally, the inclusion of goals often requires recompilation.
* *Utility functions* are mathematical utilities that evaluate the system state from measurement preferences and/or their alternatives and return a value as result which is used to enable adaptive reasoning. This approach is considered an improvement on the *goal-based adaptation* approach because it does not classify the system of states as desired or undesired but assigns a numerical value to each option, and therefore more precision in terms of applicability of the optimal option considering the current context. Utility functions enable conflict management when adaptation goals contradict each other, allow user preferences to be considered, are frequently used in QoS-based with changing environments, fit well in ubiquitous environments, are a good mechanism for systems where several self-adaptive applications are running simultaneously.
* *Case-based reasoning* is based on the assumption that similar problems have similar solutions and problems tend to recur. Chaining generalised rules CBR are used to generate adaptation decisions from prior knowledge acquired from stored cases (a large number of samples) and training, which will generate meaningful adaptation decisions. This process requires an important effort because the complexity of knowledge tasks. The simplest adaptation involves the application of rules introduced by experts or learned from the application of algorithms, on top of a previous solution. However, it is a costly process because of the knowledge required of the rules, the updates to be made and the

changes in the context. CBR can evolve because it can learn for itself from previous successful and unsuccessful decisions.

* *Reinforcement learning (RL)* is an unsupervised mechanism that selects concrete actions on a trial-and-error basis in uncertain environments to achieve a long-term goal. The immediate reward is likely to be poor. There are approaches that learn from experience and others that are somewhat slower and require more storage because they use models to explore future options before they actually occur. This approach works well in environments with uncertainty and unexpected changes, better decisions can be achieved by applying rewards to learning, and processing feedback.

– *Degree of decentralization* includes *decentralised*, *hybrid*, and *centralised.*

In addition, other classifications can be found in the literature. Kakousis et al. [125] emphasises that adaptation can be either static or dynamic. In static adaptation, elements/components are selected or customised pre-runtime (i.e., design, compilation, loading) but executed after compilation while in dynamic adaptation methods can be applied and elements/components can be modified while the system is running (i.e., at runtime) without recompiling or restarting the system. Kakousis et al. [125] comprises three categories of adaptation:

- *Requirements adaptation* includes both functional and non-functional requirements. These requirements may change because user needs change or because the functionality of the system needs to be extended. Kakousis et al. [125] categorise this as occurring pre-runtime.

- *Design time adaptation* solves mismatch between components. Kakousis et al. categorise this as occurring both pre-runtime and during system architecture analysis.

- *Dynamic adaptation* addresses the adaptation over software components to fit better the execution environment. However, the steps to be taken until the adaptation has to start are unknown. Kakousis et al. categorise this as occurring after deployment, relates it to the concept of *context-aware* system and indicates that it is often limited to non-functional aspects.

Customisation and personalisation of systems also affects different target systems. Several classifications can be found in the literature in this respect, which together include the following categorisations:

- *Component-based systems*: systems are composed of components that are reusable and can be interchanged or replaced by other components. There are component managers in charge of the reconfiguration of tasks (distributed or centralised) depending on the requirements and decisions of experts. This approach also enables the generation of executable code [190].
- *Service-based systems*: systems can be composed of services. These can be orchestrated by an orchestrator. Different mechanisms or circumstances could trigger the reconfiguration of services, e.g. a violation of a constraint.
- *Application-based systems*: executable units or applications that can be installed in heterogeneous devices.

2.2 Software and Modular Entities

Modular and computational entities encapsulate the functionality of the systems. However, while it is true that there are several alternatives that could be valid for this purpose. This section introduces each of them, presents their main characteristics, ways of operation and finally, a comparison between them.

2.2.1 Services

Service-Oriented Computing (SOC) is the computing paradigm based on services as a key *element* for applications development and systems [182]. The services are computational *elements* that can encapsulate multiple functionalities (functions can be simple or complex).

In addition, the services should be technology neutral which implies that protocols, descriptions and discovery mechanisms should conform accepted standards; related to loosely coupled, the specific knowledge or internal structure should not be required by the client or the server side; and services definitions and location information should be stored in an accessible repository in order to clients can invoke the services regardless of their locations [182].

The services can be *simple* and *composite* services. Services must implicitly maintain the quality of reuse to achieve the quality level required. Moreover, from existing services, *composite* services can be assembled. Thus, it is possible, to combine information and functions, or even integrate applications with other distributed applications to facilitate the development of systems that offer new functionalities or a more complete set of functionalities. Services can be a part of an aggregation or even be a part of the other composition of services. Usually,

the services are *business-aligned* or with the goals and they support low-cost composition of distributed applications but inter-service infrastructure (e.g., a single device, local area network, distributed or more wide area networks) is required to support service interactions and communication.

Services composition depends of *service orchestration* and *service choreography* [184]:

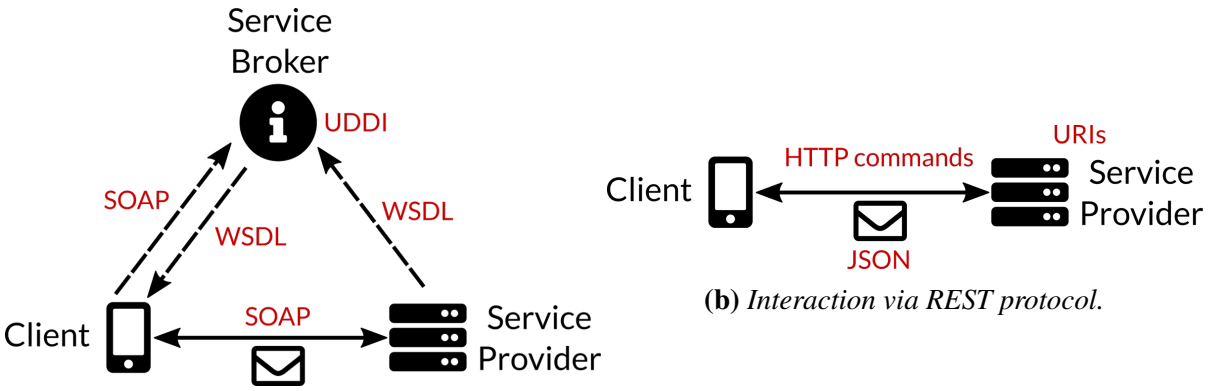
- *Service orchestration* employs a centralized approach where an orchestrator invokes (synchronous or asynchronous) and combines the services for service composition. Within orchestration process the transactions between entities must be managed.
- *Service choreography* employs a decentralized approach which involves messages exchange between two entities. Therefore, there is no central entity to control the interaction.

A *web service* is a kind of service identified by a URI which expose its description, i.e., ports, operations definition, message exchange and binding (packaging and transportation protocols are used, such as SOAP, for the interconnection) using Web Service Description Language (WSDL) as the common XML-based standard. The Universal Description, Discovery and Integration (UDDI) standard is a directory which contains the publication of the service and enables the client the location and discovering of the service [182].

Services provide uniform and ubiquitous information to a wide range of devices and software and within their specifications it is determined the granularity of replaceability to design [182]. However, they must be monitored and managed to control Quality of Service (QoS) which is related to functional, and non-functional service quality attributes such as cost, performance (e.g., response time), integrity, reliability, scalability, and availability [132]. At this regards, it is should be noted Simple Object Access Protocol (SOAP) and Representational State Transfer (REST) as key elements in the development of services [250, 146]:

- *SOAP* protocol is designed to support the exchange of information from programs developed on different platforms or in different programming languages. It uses WSDL (based on XML format) to expose to the client the operations that can be performed by the service.
- *REST* is an architectural style [98] designed mainly to operate with components and objects. It uses URI to invoke create, read, update and delete (CRUD) or GET, POST, PUT, and DELETE operations, supporting several formats (text plain, XML, HTML and JSON).

REST emerged after SOAP. REST can make use of SOAP but not vice versa. SOAP is better suited for situations where the state of information should be maintained and SOAP 1.2 provides additional features to improve security and reliability. However, SOAP messages contain a significant amount of information and more bandwidth is needed than with REST. Therefore, REST needs less bandwidth than SOAP, which it is relevant in instances where there are constraint on network bandwidth [250, 146]. An illustrative interaction with both communication mechanisms, SOA and REST can be seen in Figure 2.4.



(a) *Interaction via SOAP protocol.*

(b) *Interaction via REST protocol.*

Fig. 2.4 *Service communication mechanisms (own elaboration).*

2.2.2 Microservices

Thönes defines a *microservice* as *"small application that can be deployed independently, scaled independently, and tested independently and that has a single responsibility"*, considering as a single responsibility as that there is *"a single reason to change and/or a single reason to be replaced"* and that it does *"one thing alone"* (i.e., functional requirements or non-funcional, even cross-functional) and "can be easily understood" [237].

Nadareishvili et al. [170] defines a *microservice* as *"small, autonomous services that work together"* that must do one thing well. It also notes that they are characterised by being a separate entity that maintains a low coupling and communication between them must be through network calls, that they are deployed independently and possibly autonomously, that consumers are not necessary, that they allow and encourage the use of different technologies (e.g., to improve performance) for the benefit of the system, that they are highly scalable, scalable even on demand, and that they enable and encourage parts of the system to be run on devices with low computational capabilities. In addition, they support the development of solutions decentralised and with speed, their replacement or elimination is optimised as they are smaller

pieces, their cost is lower and it is a way to ensure resilience (i.e., if one component fails, no further cascading failures occur) [170, 164].

The microservice architecture model can be seen in Figure 2.5. Here are highlights of some of the benefits it provides [164]:

- *Agility* by providing quick and guided solutions.
- *Composability* favouring reuse and minimising development times.
- *Comprehensibility* with a focus on simplicity and accuracy.
- *Independent deployability* encourages flexibility and the acquisition of new features by certain components.
- *Organizational alignment* of the working group favours the provision of more complex solutions.
- *Polyglotism* on options and technologies.
- *Efficiency* increases by reducing the cost of infrastructure which is related with *independent manageability* and makes stakeholders have less need to intervene.
- *Resilience* because it increases availability and the user experience is better.
- *Tests* carry fewer risks.

Microservices are a solution for legacy architectures because they provide a *"more granular and modular level"* of systems and applications simplifying the complexity of the problem and providing on-demand services, both of which improve performance [71].

2.2.3 MultiAgents

Wooldridge defines the *MultiAgent Systems (MAS)* [262] as those *"composed of multiple interacting computing elements known as agents"* being the *agents "computer systems with two capabilities"*:

- Agents can take action autonomously by deciding what to do to achieve their objectives.
- Agents can cooperate, coordinate and negotiate with other agents, i.e. conduct interaction beyond a simple exchange of data.

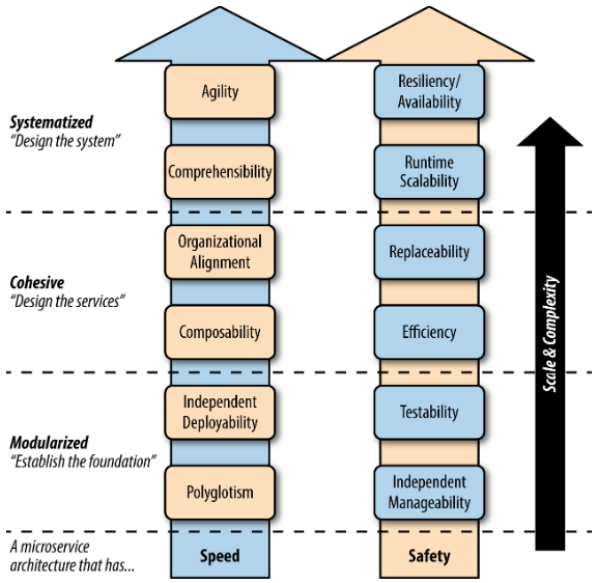


Fig. 2.5 *Microservice architecture model advantages proposed by Nadareishvili et al. [164]*

An agent can be constructed from the composition and interaction of modules, which provide responses from the inputs (from sensors) and state of the agent itself [143].

Intelligent agents [143] are:

- *Reactive* because they are able to perceive their environment react accordingly.
- *Proactive* because they exhibit *"goal-directed behaviour"*.
- *Socially skilled* because they are able to interact with other agents to achieve the designed goals.
- *Autonomous*, i.e. each agent has control over its state and behaviour and they can act in a dynamic and unstructured environment.
- *Selfish* because there is no guarantee that an agent i will execute an action α of another agent j that is not in his or her (i) best interest.

Agents have the ability to coordinate their actions with each other through a blackboard, without explicit communication, but explicit communication simplifies interaction. They should use a common language in communication [80]. Speech between agents are based on the five types identified by Searle et al. [210] that are *"representatives"*, *"directives"*, *"commissives"*, *"expressives"* and *"declarations"*. The ARPA consortium developed the first support for agent communication called Knowledge Sharing Effort (KSE), which facilitated the sharing and reuse of both, knowledge bases and systems. The KSE contemplates two languages:

- The Knowledge Interchange Format (KIF) designed as a language to express message content. With KIF, agents [262] can express:
  - *"Properties of things in a domain"* such as *"Michael is a vegetarian"*.
  - *"Relationships between things in a domain"* such as *"Michael and Janine are married"*.
  - *"General properties of a domain"* such as *"everybody has a mother"*.
  - *"Relation between two objects"*.
  - *"Definition of new concept"*.
  - *"Relationship between individuals in the domain"*.

  KIF assumes a basic, fixed logical apparatus, which contains the usual connectives that one finds in first-order logic [262].

- The Knowledge Query and Manipulation Language (KQML) [262, 80] which is a message format and a protocol to share knowledge at runtime. It defines a set of acts of communication or performatives that can express:
  - *Achieve* such as *"S wants R to do make something true of their environment"*.
  - *Advertise* such as *"S is particularly suited to processing a performative"*.
  - *Ask-about* such as *"S wants all relevant sentences in R s VKB"*.
  - *Ask-all* such as *"S wants all of R s answers to a question"*.
  - *Broadcast* such as *"S wants R to send a performative over all connections"*.
  - *Error* such as *"S considers R s earlier message to be malformed"*.

  There are more performative ones, this is just a sample. Also noteworthy is the entity of *communication facilitators* who coordinate the interaction between agents (i.e., they acts as *"agent-servers"*) [80].

- FIPA ACL (Agent Communication Language) is similar to KQML [262]. The main difference is that the knowledge can be expressed in different languages (e.g, KIF, Prolog, etc.). Some examples of performatives are:
  - *Query* such as *"Is the door open?"*.
  - *Agree* such as *"OK! I'll open the door"*.

- *Inform* such as *"The door is open"*.
- *Failure* such as *"I am unable to open the door"*.
- *Subscribe* such as *"Say when the door becomes open"*.

2.2.4 Capabilities Comparison of the Software and Modular Entities

Here, a comparison (Table 2.1) of the capabilities of the software and modular entities studied has been drawn up.

Table 2.1 *Capabilities Comparison of the MultiAgent Systems (MAS), Service-Oriented Computing (SOC), and Self-Adaptive Services (own elaboration).*

MultiAgent Systems (MAS)	**Service-Oriented Computing (SOC)**	**Self-Adaptive Services**
Approaches to build complex software systems		
Autonomy		
Agents act individually fulfilling their individual goals	No inter-service dependencies	
Adaptive		**+ Adaptability**
in response to changing requirements, services and exceptions		
Sociability	**Interoperability**	
Intelligent society	Interface level	
Communicating agreed-upon protocols		
Distributed	**Distributed/Centralised**	
There is no a central control entity	Choreography/Orchestration	
Rationality	**Encapsulation**	
Reactivity		**+ Reactivity**
Proactivity	**Availability/Discovery**	
Highly formal specification	Public registries	

It can be observed that MAS and SOC are different approaches that share multiple similarities. Therefore, agents and services could be interchangeable entities but there are two points of view that underline their differences [179]:

- Agents could be deployed as a service and any service could be conceived as an agent.
- Services are simpler things that agents. The first ones could be used by the second ones (as customers) because agents can support services. Note that *"while, in principle, autonomic functionalities can thus be implemented by using agents to create high-level behaviours coordinating the system's functional layer, it is important to provide the latter with proper interface and reflective functionalities"* [179].

It should also be noted that self-adaptive services complements SOC adding reactivity and adaptability capabilities.

2.3 Software Architectures

Bass et al. point out that architectures as a *"crucial part of the design process"* and define *software architecture* as *"the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them"* [29]. Architectures can favour the construction of systems by adjusting the goals, in terms of efficiency, providing more reliable systems, in less time, and more economically [29].

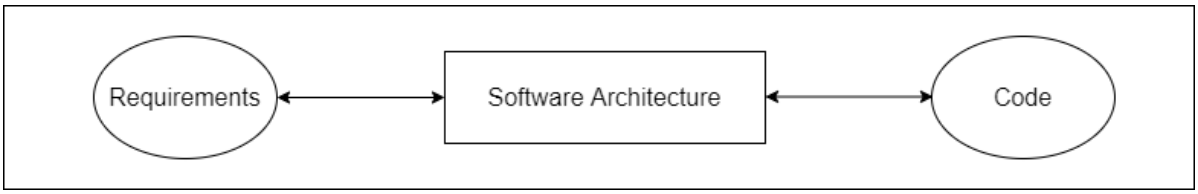


Fig. 2.6 *Software architecture connects requirements and code proposed by Garlan [87]*

Software architectures serve as connector between requirements and implementation (Figure 2.6) and encompass six key aspects related to system development [87]:

- *Understanding* refers to that the architectural descriptions present the systems from a higher level of abstraction, facilitating their understanding and capturing their representation.
- *Reuse* at different levels, e.g., components included into architectures or frameworks.
- *Construction* in terms of components and the relationships between them.
- *Evolution* of the systems encompasses addresses, branches, edges, and cost estimations. Usually the functionality of the components is separated from the interaction mechanisms between these components, the latter referring to the way in which the components are connected. Such a separation provides flexibility so that the mechanisms can be replaced.
- *Analysis* refers to consistency, style, quality attributes or dependencies inter alia.
- *Management* during the development process.

2.3.1 Service-Oriented Architecture (SOA), SOA 2.0, and Event-Driven Architecture (EDA)

Service-Oriented Architecture (SOA) can be defined as *"an enterprise-scale IT architecture for linking resources on demand"* which comprises a set of *"business-aligned"* services that

can be choreographed into composite applications and collectively fulfill the business process and goals of the organization [14]. IBM defines SOA as *"a way to make software components reusable via service interfaces. These interfaces utilize common communication standards in such a way that they can be rapidly incorporated into new applications without having to perform deep integration each time."* [111]. Its features [147] include:

- *Low coupling* because services must be *"invoked independently of their technology and location"*.
- *"One-to-one communications"* because a service is *"invoked by one consumer at a time"*.
- *"Consumer-based trigger"* because *"the flow of control is initiated by the customer (the service consumer)"*.

Fig. 2.7 *Request/reply mechanism in SOA proposed by Maréchaux [147]*

Each service design of SOA should comply the eight following principles [132, 75]:

- *Loose coupling* thereby minimising the dependence on one another. Therefore, if the service changes internally, the client would not be affected.
- *Abstraction* meaning that the client can know what the service does but not how, i.e. the details of the execution of the service's functionality are not exposed.
- *Reusability*, the service should able to be used in different applications, therefore it is recommended to have a modular design.
- *Autonomy* may refer to *runtime* or *design-time autonomy*. *Runtime autonomy* refers to *"the level of control a service has over its processing logic at the time the service is invoked and executing"* while *design autonomy* is defined as *"the level of freedom [service owners] [. . .] have to make changes to a service over its lifetime"* [75].
- *Statelessness*, services should not maintain information on their previous status.

- *Discoverability*, services should be registered and service information is kept in this register.
- *Composability*, services must be modular and independent and provide answers to simple problems, and all this favours their combinability to provide solutions to more complex problems.
- *Interoperability* through the use of standards to support communication and interaction between suppliers and clients.

Moreover, ideally, services should support a dynamic reconfigurable style [14] and their infrastructure supports restructuring and upgrading services on demand [185].

SOA also defines the interaction between agents that can be both, *clients*/*consumers* and *providers*, that involve the *publish*, *find* and *bind* operations. *Providers* publish a description of the service(s) that they provide, and *clients* must to be able to find the description(s) to request the execution of the service (Figure 2.7). The procedure is similar to client-server architecture but it is also one of the primary SOA goals, specifically to allow the access to remote interface of components through request-response (e.g., Remote Procedure Call (RPC)) methods because new Information Technology (IT) systems required to be not only reactive but also proactive [45].

Services are flexible because the separation of the interface (*service usage interface* is the interface exposed to the clients), implementation, and binding, and because they make it possible to defer the choice of service provider to a given point in time on the basis of new functional and non-functional (i.e., scalability, performance) requirements [14].

The SOA architecture (Figure 2.8) comprises the following layers: within the Operational System (layer 1), the applications can be packaged and custom, and system integration using *service-oriented integration* is possible; Components (layer 2) comprise domains, goals, processes and criteria of the component election by clients, functionality and QoS; in Services (layer 3), services description is exposured, then they can be discovered and invoked; in Composition and Choreography (layer 4), the services exposed in layer 3 can be orchestrated and choreographed, and be bundled into a single application (for a specific use case) or system; the Presentation (layer 5) involves several standards and technologies such as Web Services for Remote Portlets (WSRP) that allow *plug-n-play* visual, user-facing web services with web applications; Integration of services comprises a reliable set of capabilities, WSDL for binding that include information on where the service is located and mechanism for integration; and QoS there are *sense-and-respond* mechanisms, tools, standards implementations and protocols to monitor the health of SOA applications and control their quality.

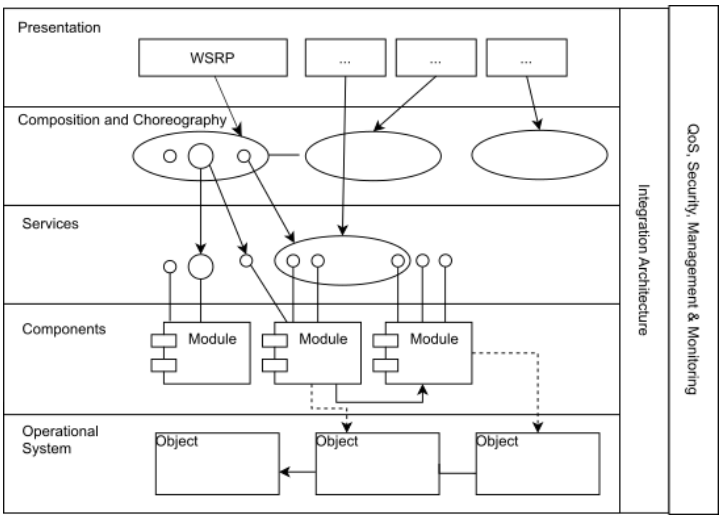


Fig. 2.8 *The layers of SOA proposed by Arsanjani [14]*

SOA 2.0 arises as a combination of SOA and Event-Driven Architecture (EDA). SOA and EDA are complementary and specifically, it consists of the interaction between events and services [155].

An event can be considered as an occurrence of a *thing* or *fact* that can be detected at a given point in time [85]. It could represent, for example, a problem, a condition, a threshold inter alia [155]. There are several events-types (e.g, time events, transaction events, atomic events, etc.) [85]. Events are also characterised because can occur independently of each other but some can cause other events [141]. Each event has a header which contains the event specification that describe its occurrence (i.e., ID, type, name, timestamp, number of occurrence, and creator) and a body which comprises a description about about the occurrence [155].

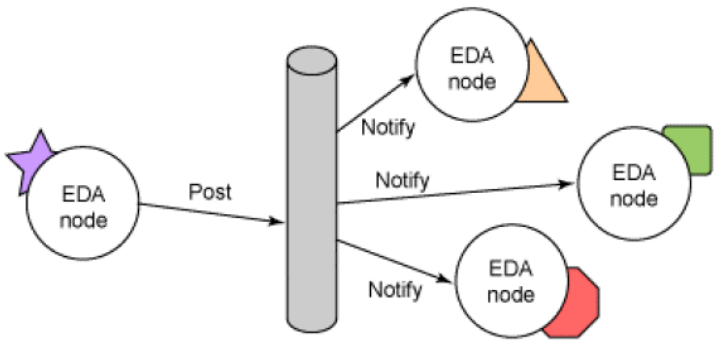


Fig. 2.9 *Publish/subscribe mechanism in EDA proposed by Maréchaux [147]*

EDA publish events without relying on the availability of a publisher, its interaction can be seen in Figure 2.9 and its features [147] include:

- *Decoupled interactions* because publishers *"are not aware of the existence"* of subscribers.
- *Many-to-many communications* because each event can have many subscribers.
- *Event-based trigger* refers to the flow going to the subscriber.
- *Asynchronous* operations are supported.

In SOA and EDA two interactions [155] can be distinguished:

- *Event-Driven SOA* refers to when an event may deploy one or more services.
- *Service as Event Generator* refers to when an event is evaluated which could be carried out within a service, and then an action is performed. In other words, a service can generate an event.

2.3.2 Resources Oriented Architecture (ROA)

Resources Oriented Architecture (ROA) [98] is the structural design of specific resources or technologies applied mainly within the Web of Things. Resources refer from physical objects (e.g., sensors or wearables) but it is not only an IT infrastructure but also objects, states, or even transactions. ROA is considered as a set of guidelines to implement a RESTful architecture. Each resource is accessed by an Uniform resource Identifier (URI) via Hypertext Transfer Protocol (HTTP) interface which contains GET, PUT, POST and DELETE operations. Figure 2.10 shows an example of interaction between gateway and Internet.

ROA is less general than SOA because it depends on the technical architecture of the services where HTTP protocol is used.

2.3.3 Agent Architecture

"An agent architecture is a software architecture intended to support [...]" an *"action selection"* within a *"decision-making process"* [262]. Usually also referred to as Belief/Desire/Intention (BDI) Architecture. Beliefs being the knowledge that the agent has about the world based on his or her perception; desires being the goals he or she intends to achieve; and intentions being the choices to be made and make up the plan [42].

Bratman et al. [42] proposes an architecture to represent Beliefs, Desires and Intentions (BDI) of the agents considering that:

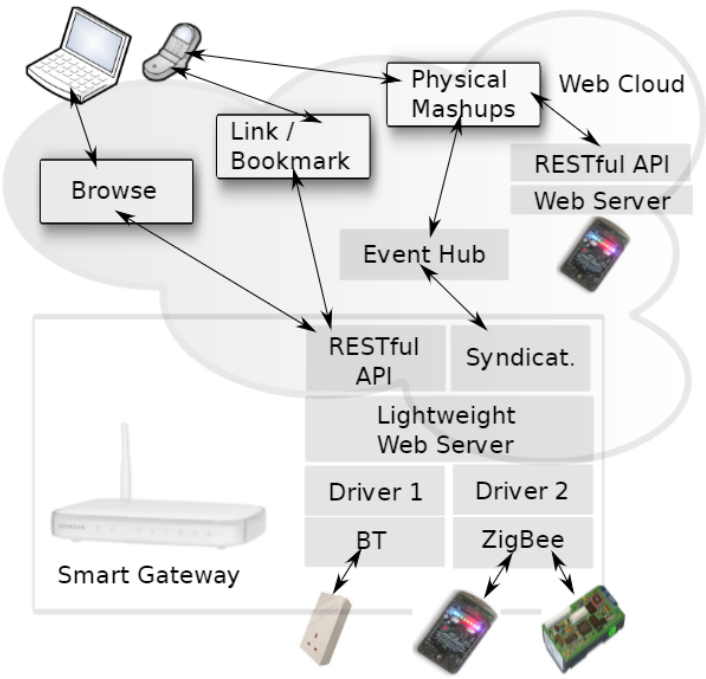


Fig. 2.10 *Gateway interaction with Internet (included Web) proposed by Guinard et al. [98]*

- the construction of an action plan requires a time-consuming;
- that the construction of this plan is based on known conditions but that these may change in the future;
- it analyses alternatives and interactions;
- it must address the problem of resource constraints.

A specification of the reasoning component of an agent is presented in Figure 2.11 [42]. The *intentions* are part of a larger plan being a plan the behaviour to be followed by the agent in order for it to achieve its own desires. From *perception* beliefs or desired can be updated in each iteration.

The *plan library* represents beliefs about the actions to be taken in the current plan to achieve what is expected. The reasoning part is carried out by *means-end reasoner*, *opportunity analyzer*, *filtering process* and the *deliberation process*. Specifically, *filter override mechanism* sensitivities the environment to suspend part of certain plans and consider other options. The *compability filter* operates in parallel with *filter override mechanism* but even if an option does not pass the filter it can still be chosen. The *deliberation process* decides which option to choose, between incompatible or a previously considered action satisfying a prior intention.

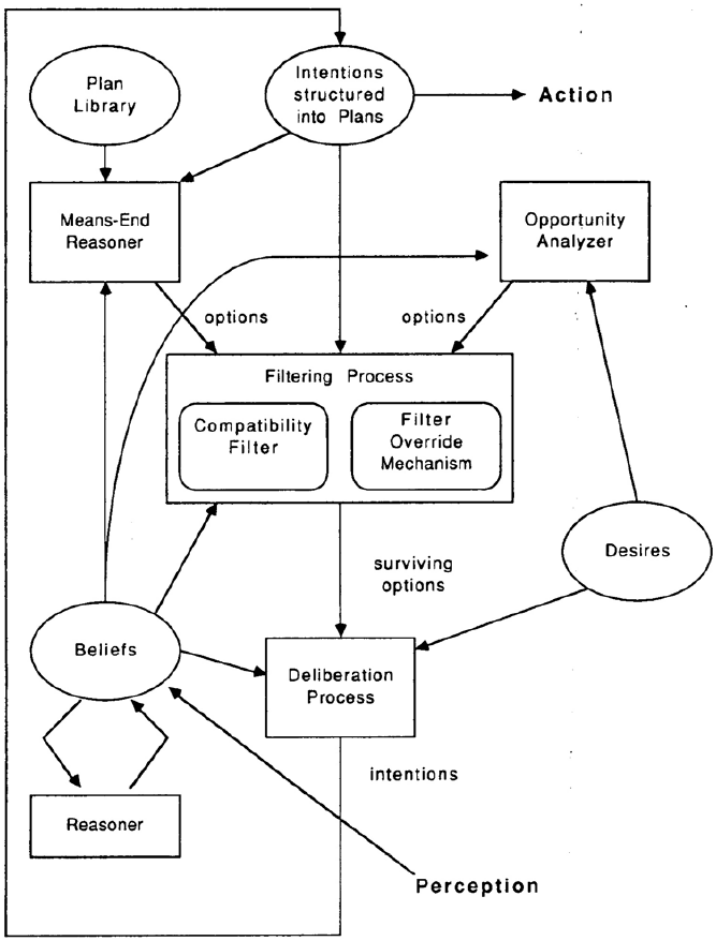


Fig. 2.11 *Practical reasoning component of an agent proposed by Bratman et al. [42]*

2.3.4 Domain-Specific Software Architecture (DSSA)

Tracz et al. note that a domain refers to a common applications or problem to solve, while a domain model is the representation of it (e.g., concepts, taxonomies, ontologies, etc.) through terms, objects and the relations between them [241, 103]. A software architecture can accommodate components, connections and constraints, with components being elements or modules that have an interface. Therefore, a domain-specific software architecture (DSSA) [241, 103] comprises requirements, domain model, supporting infrastructure and processes. The diagram of an architecture also is made up of components, alternative components, data, connectors, and constraints. Its representation can be graphical and textual. A template can be followed as a guide and variability is supported by parameterisation or through interfaces [241]. In DSSA, each domain can have several architectures of different styles being the client-server architecture style only one of the possibilities of DSSA. Tracz et al. [241] denote object-oriented design as a technique related to Domain Model.

The evaluation of an DSSA architecture [241, 103] is based on the constituent components, ease of use, coverage of the application domain, productivity and cost, and then on the ability to extend or adapt the architecture. Moreover, applications can be compared to validate the architecture.

2.3.5 Reference Architectures for Adaptation and Self-Adaptation

Several researchers endorse the use of architectures as mechanisms for adaptation and self-adaptation to handle a wide variety of systems. In general, such mechanisms include an iterative operational process, as a part of a more complex whole, which distinguishes mainly three phases: context detection, reasoning based on that context considering the different options, and acting [125]; or MAPE-K control loop composed mainly of four activities: Monitor, Analysis, Plan and Execute that share knowledge among them [127]. However, there is a need to reduce the cost added for including external control to the systems. Proposals may include architectural models in order to provide an overall view of the system, component composition, interconnections, properties, integrity constraints, control loops, and computational reflection [88]. In this regards, four of the most outstanding architectures that have served as a reference in research and industry are presented here.

MAPE-K

Autonomous systems possess collections of autonomous elements. Those elements and their external environment are controlled which makes it possible their management by an autonomic manager. This is the basis followed by most self-adaptive systems approaches, which usually base their designs on control cycles [44]. Figure 2.12 shows the reference feedback control loop known as MAPE-K [127] proposed by IBM. At the different phases, the autonomic manager monitors (M) by sensors, to subsequently analyse (A) the data, to then compose a plan (P) based on a set of actions and execute (E) them via actuators. In addition, there is a knowledge (K) component to share constraints, encode rules and to consider the knowledge acquired in the other four phases [127].

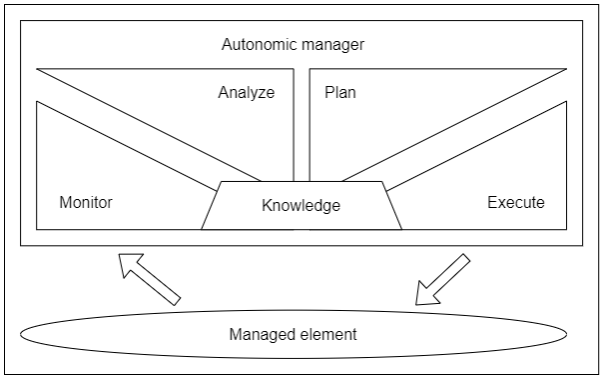


Fig. 2.12 *MAPE-K reference self-adaptive control loop proposed by Kephart et al. [127]*

However, there are three main limitations that should be mentioned: 1) the interaction between autonomic managers is not modelled and then the proposal is limited to centralised applications; 2) the modelling of the knowledge component is not granular, and the possibility of applying reasoning about adaptation across multiple dimensions is not considered; and 3) the knowledge encoded is limited, and therefore the system can not evolve.

Rainbow

Garlan et al. present Rainbow [88] framework to support self-adaptation of systems using software architectures and a reusable infrastructure. Figure 2.13 shows Rainbow control loop that models the detection of constraint violation (i.e., a composition of the elements not allowed) based on monitoring and evaluation, and the possibility of global or modular adaptation. The infrastructure is divided in the three following layers: 1) *System* layer includes the access interface, a measurement mechanism (called *Probes*), a *Resource discovery* mechanism and the *Effectors* mechanism to performs the modifications; 2) in *Architecture* layer, *Gauges* adds

information to *Probes* and update properties. The *model manager* manages the access to the architectural model that is evaluated by the *Constraint evaluator* which checks the model and triggers the adaptation to be carried out by the *Adaptation engine*; and 3) *translation* layer maps the information as intermediary (e.g., an architectural change into a system change).

The components (i.e., computational elements, data stores, clients, servers, databases, interfaces, etc.) are represented graphically as nodes and connectors reflect the interaction between them. Moreover, computational elements must provide for expected throughputs, latencies, and protocols [88].

Rainbow is a generic proposal to address requirements of a variety of applications from different domains. The layered layout allows independent changes to be made. However, it has limitations in terms of: 1) remaining a centralised approach which impacts on scalability and localised failures; 2) to be extended, e.g. by injecting human-defined strategies; 3) it is not proactive, as it does not anticipate adaptation, but implements it after detecting this need; and 4) response time architecture is not assessed.

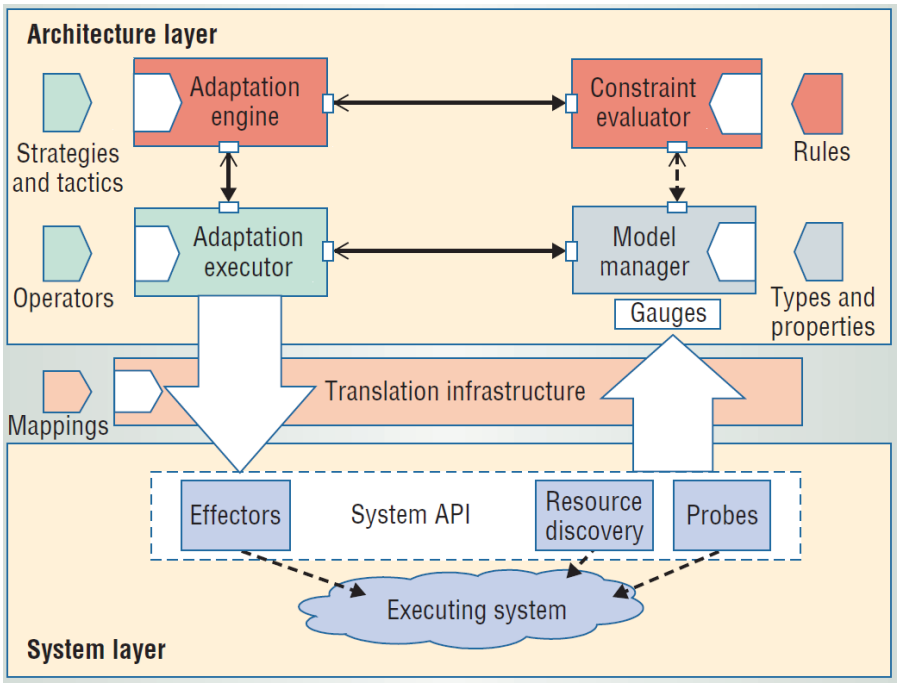


Fig. 2.13 *Rainbow architecture based self-adaptation with reusable infrastructure proposed by Garlan et al.[88]*

Reference Architecture for Decentralised Self-Adaptation

Weyns et al. [260] present a decentralised architecture where there is not a central point of control, for distributed self-adaptive software systems. Any of the phases (i.e., monitoring, analysis, planning, and execution) may lack a control point, which implies the unavailability of global knowledge at any host.

It is necessary to establish a mechanism to coordinate the different entities of the self-adaptive units:

- *Coordinated monitoring* collects data of its underlying system and optionally of its environment. These data must be shared and synchronised locally.
- *Coordinated analysis* to obtain a more complete knowledge of the system, and thus identify violations and predictions. It should be noted that in centralised systems, each unit has all the data to do the analysis (its own data), but in decentralised systems, each unit has only a part of the data (its own data).
- *Coordinated planning* coordinates the goals of the different systems units. In centralised systems, each unit may have its own goals, may behave selfishly, and there may be conflicts between goals. Therefore, a plan is required to coordinate adaptation and cover different goals.
- *Coordinated execution* executes the orders to carried out the changes of the adaptation. In centralised systems, each unit has control over the order and timing in which the adaptation is executed. However, in decentralised systems, the adaptation has an impact on the whole system and changes have to be synchronised.

Figure 2.14 shows the proposed model that brings together the concepts and methodology of MAPE, feedback-control loop, computational reflection, and several components for decentralisation such as *Local Managed System*, *Self-Adaptive Units*, *Meta-Level Computations* for adapting the local managed system from a set of *Meta-Level Models*. The latter is the generalisation of four entities. 1) *System Model* represents the system or parts of the system that can be local managed or self-adaptive unit; 2) *Concern Model* are the goals that can be represented as rules (e.g., event, condition, action); 3) *Working Model* is the data structure of the specific domain, such as the temporary representation of possible redeployment architectures proposed by an analyser and their instructions all produced by an analyser; and 4) *Coordination Model* represents the data required by a meta-level computation to coordinate with others meta-level computations of other units.

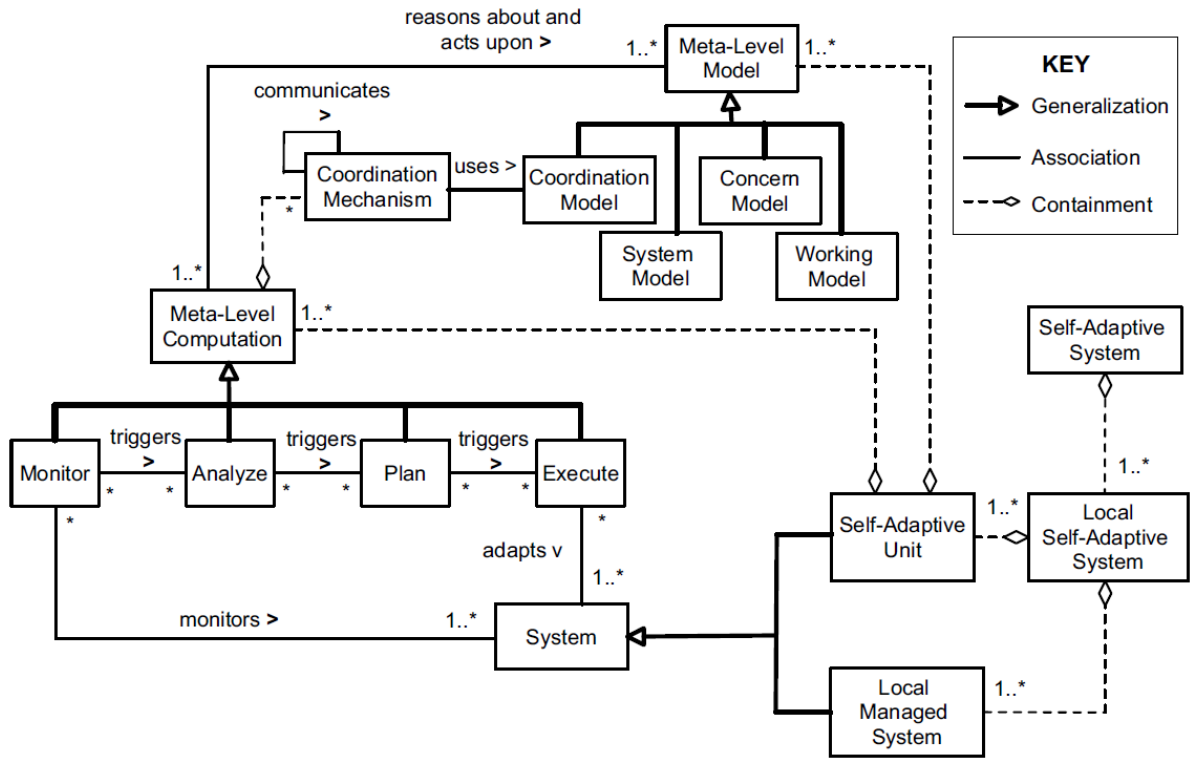


Fig. 2.14 *Reference architecture for decentralised self-adaptation proposed by Weyns et al. [260]*

The proposal is based on the development of two case studies in which the authors monitor traffic by deploying a set of cameras. There is a collaboration between cameras to detect certain patterns in traffic and it also discusses how to detect when a camera fails and how to perform its recovery based on the rules mechanism.

This architecture for decentralised self-adaptation includes *Meta-Level Model* and *Meta-Level Model* to facilitate the adaptation and coordination between different self-adaptive units. Here, system scalability is possible. However, it has limitations in terms of: 1) No specific mechanisms for resolving conflicts between concerns are included; and 2) it has not been evaluated either theoretically or practically with simulations or in a real environment and then response time architecture is not assessed.

Dynamic Adaptive, Monitoring and Control Objectives model (DYNAMICO)

Villegas et al. present DYNAMIC [247] as a reference model for context-based self-adaptive software which includes three adaptation levels. Its definition includes the elements and their functionalities, the control and data interactions, one independent loop for each level and interactions between loops. Functional and non-functional requirements, and adaptation properties are defined as variables that can be controlled (called *control objectives* and *adaptation goals*). All of them must be satisfied by the target system.

- *Control Objectives Feedback Loop (CO-FL)* manages changes in adaptation goals and user requirements. This operates with the Service Level Agreements (SLAs) that usually represents QoS and Service Level Objective (SLO) that represent performance conditions. The control objectives can be modified by user at runtime, therefore consistency and synchronisation must be managed during adaptation and those changes derive within *Adaptation Feedback Loop* and *Monitoring Feedback Loop*.

- *Adaptation Feedback Loop (A-FL)* performs the adaptation in accordance with the control objectives. It measures the variable errors (e.g., a control objective is not met) and their inputs and to report signs of adaptation to the A-FL analyser. The A-FL analyser decides whether adaptation is necessary.

- *Monitoring Feedback Loop (M-FL)* analyses context signs and facts to support the adaptation and the control of objectives. The context inputs are derived from CO-FL control objectives. It generates an adaptation plan runtime, from a semantic Web inference rules part of a Smarter-Context ontology, by modifying existing sensors and monitoring conditions and deploying a new strategy of monitoring.

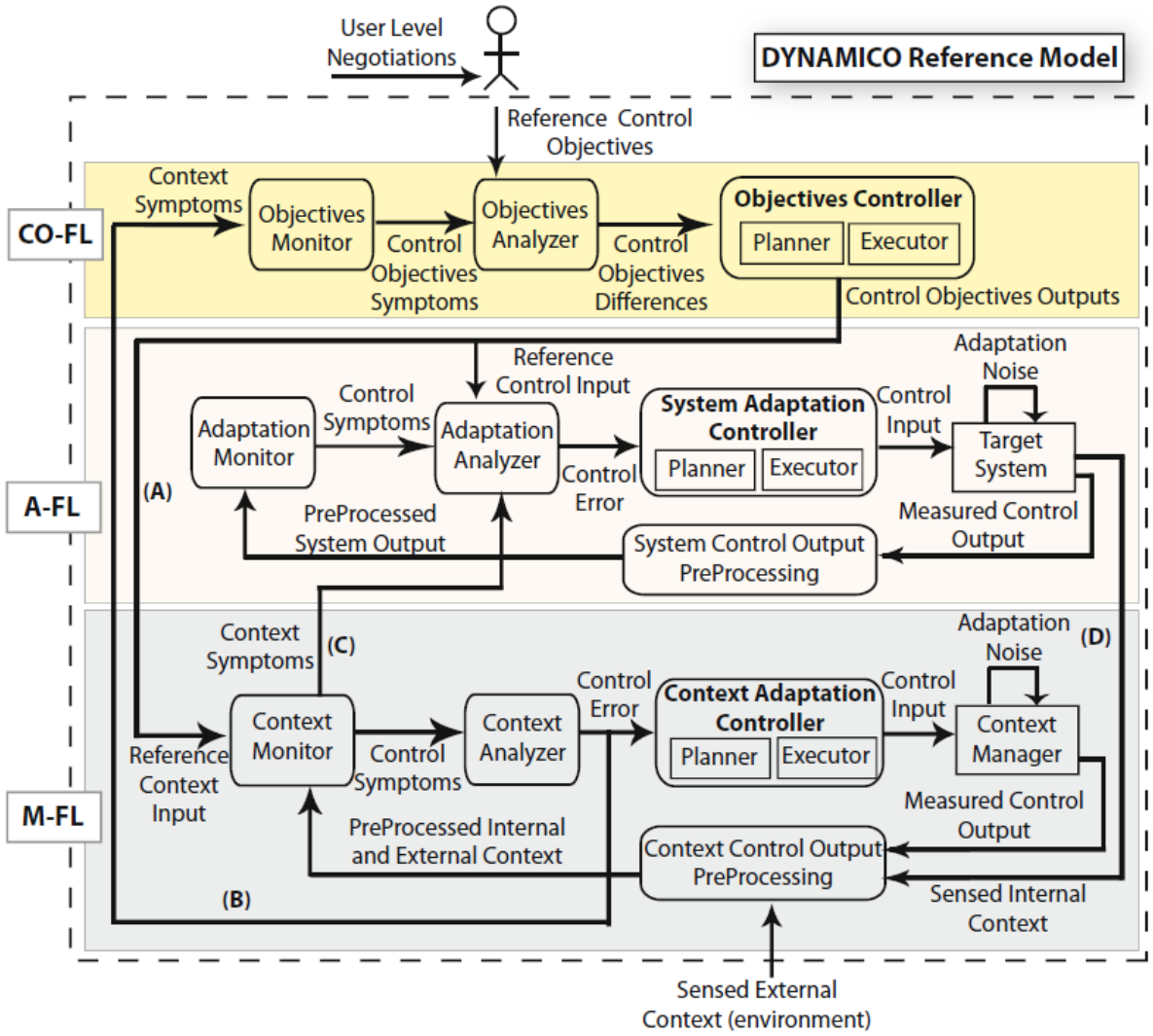


Fig. 2.15 *DYNAMICO reference model for self-adaptative systems proposed by Villegas et al. [247]*

Figure 2.15 shows the proposed model that brings together the three levels described above with their corresponding loops, and controllers. However, DYNAMIC has limitations in terms of: 1) dynamic discovery; 2) decentralised adaptation controlled by distributed feedback loops; and 3) it has not been evaluated either theoretically or practically with simulations or in a real environment and then response time architecture is not assessed.

2.4 Frameworks

McIlroy introduced the concept of reuse in 1968 by presenting a library of components that could be reused and customised, achieving different levels of precision and robustness [150]. Software reuse has become a *"standard practice for software construction"* where existing software artefacts are used to create new software systems [134]. The term artefact encompasses design and implementation of structures, specifications, transformations, code fragments inter alia [134]. In this regards, techniques of reuse come in the form of component libraries, application generators, code compilers or even templates based on the abstraction, selection, specialisation (on the basis of parameterised specialisation, transformations or constraints) and integration (exporting functions from where they are implemented and importing them into modules where they are to be used).

Frameworks share multiple characteristics with reuse techniques in general and in particular. Johnson et al., in 1988, defined a framework as *"a set of classes that embodies an abstract design for solutions to a family of related problems, and supports reuses at a larger granularity than classes"* [122]. Later, in 1992, he defined them as *"a reusable design of a program or a part of a program expressed as a set of classes"* and how these classes collaborate, and alternatively, as *"a reusable design for solutions to problems in some particular problem domain"* [120]. A framework comprises code, blocks for build a software systems or subsystems but also abstract designs [120, 215]. *"Frameworks are designed by experts in a particular domain and then used by non-experts"*[120].

Johnson et al. also highlight that *"one of the most important kinds of reuse is reuse of designs [...] The design of a program is usually described in terms of the program's components and the way they interact"* [122]. In fact, frameworks *"are an object-oriented reuse technique"* [121]. In particular, their definitions [121] refer to:

- its *structure*, when a *"framework is a reusable design of all or part of a system that is represented by a set of abstract classes and the way their instances interact"*; and

- its *purpose*, when *"a framework is the skeleton of an application that can be customized by an application developer."*

Initially, the reuse techniques were based on components, in order to connect these components to build new systems. Frameworks make development tasks easier for developers who do not have to know how these components are implemented, only how to use them to customise systems. In addition, frameworks make the resulting systems more efficient, easier to maintain and reliable [121]. The frameworks [121]:

- *"are more like techniques that reuse both design and code"*;
- they are *"easier to extend and combine than special purpose languages"*; and
- frameworks also are *"a kind of domain-specific architecture". "The main difference between them is that a framework is ultimately an object-oriented design, while a domain-specific architecture might not be"* [121].

Frameworks encompass different approaches and technologies for reusability being *reusability "a key to improving software development productivity and quality"* [33]. Biggerstaff et al. proposes a characterisation that can be seen in Figure 2.16 and mainly differentiate between compounding and generation technologies [33]. Composition is based on the idea that components are atomic and that composition principles can be applied (message passing for function calls or static bindings, and inheritance allowing dynamic method invocations). Within generation technologies are code patterns and transformation rules. Reusability implies finding a component, understanding it, modifying it and composing it. Modules also become a part of reuse by encoding specific information [33].

A pattern *"describes a problem to be solved, a solution, and the technique in which that solution works"*. Patterns help to solve permanent recurring problems over time, being the *design patterns "the micro-architectural elements of frameworks"* [121]. The purpose of the framework can be described with patterns, where a pattern is a description of a problem that occurs repeatedly in the domain of the framework together with a description of its solution [120]. Patterns serve as a guide for the assembly of the framework components [94] (e.g., Model View Controller, MVC).

Frameworks *"are firmly in the middle of reuse techniques. They are more abstract and flexible than components, but more concrete and easier to reuse than a pure design (but less flexible and less likely to be applicable). Although they can be thought of as a more concrete form of a pattern, frameworks are more like techniques that reuse both design and code,*

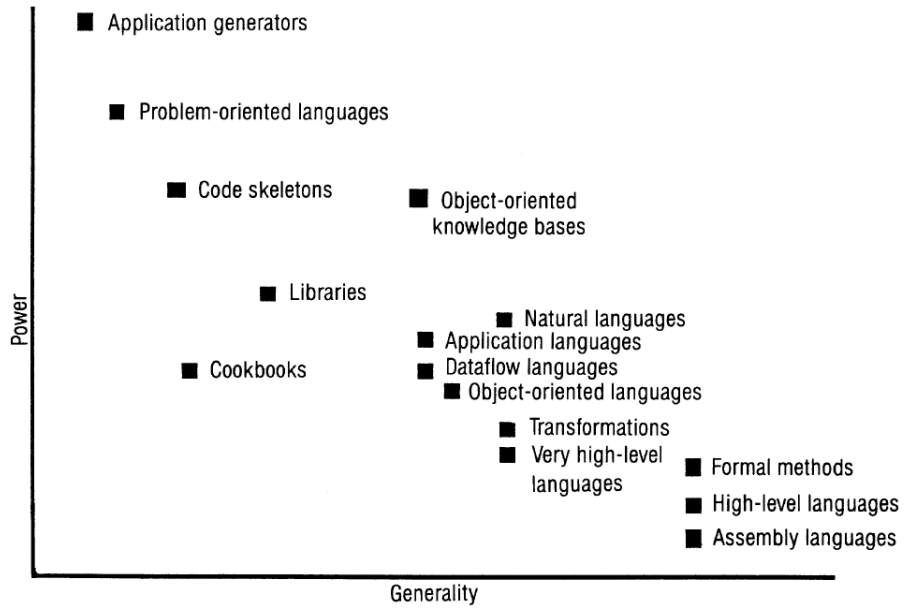


Fig. 2.16 *Reusability technologies characterisation proposed by Biggerstaff et al. [33]*

such as application generators and templates" [121]. A framework can reuse code ensuring requirement consistency [122]. They can be used by application developers with no knowledge of how it works internally but it can also show them several design details that are embodied within it [120].

The frameworks can be built on top of another framework, it can be interlocked with other frameworks (e.g., sharing abstract classes and specializations) [122]. Even the development of applications often requires the integration of different frameworks such as GUIs, communication systems, databases, etc, in conjunction with existing libraries, legacy and components [77].

Fayad et al. defines the object-oriented application frameworks as a technology used for *"reifying proven software designs and implementations in order to reduce the cost and improve the quality of software"* [77]. A framework is characterised by its modularity, reusability, extensibility, inversion of control and for favouring the reuse of components. In particular, a framework can also act as a coordinator of a sequence of activities by inversion of control, resulting in an extensible skeleton [122]. From *hook methods*, the frameworks introduce the required variations in a particular instantiation [77]. A framework can involve several basic abstractions, such as *components* representing the elements in a domain and encapsulating the behaviour of the objects; *tools* for the manipulation of the components through visual effects; *commands* defining the operations of the components, in a similar way to the exchange of messages with the difference that they are also treated as objects; and *external representations* transmit information outside the application [248].

Software frameworks [215, 149] follow the *"Hollywood Principle: Don't call us, we'll call you."*. These software frameworks contain *frozen* and *host spots* [215, 149]. On the one hand, frozen spots are fragments of code already implemented within the framework that refers to common aspects of the domain-specific application architectures (i.e., components and relationships). These do not change when an instantiation takes place. On the other hand, hot spots have a generic design and are specialised to suit the needs of the application under development.

Fayad et al. [77] propose the following classification of frameworks based on their objectives:

- *System infrastructure frameworks* simplify the development of system infrastructure (e.g., operating system), communication, interfaces, and language processing tools.
- *Middleware integration frameworks* integrate distributed applications, components, favouring reuse, modularisation and how to extend the software infrastructure in a distributed environment.
- *Enterprise application frameworks* support *"the development of end-user product applications and products directly."*

There are *compiler frameworks* which include both table classes used for the instantiation of objects and classes to generate code, or framework for construction of user interfaces [122].

For their side, Rubensteint et al. [196] outline the following classification of frameworks:

- *Conceptual framework* considers problem solving in its entirety, with a need to determine which parts are part of the system and which parts are part of its environment. Rudestam et al. note that a *conceptual framework "is simply a less developed form of a theory, consists of statements that link abstract concepts to empirical data"* [197]. More recently, Weible et al. [256] identify within a *conceptual framework "a set of variables and the relationships among them that presumably account for a set of phenomena"*
- *Theoretical framework* adopts system thinking which is a conceptual framework. A *theoretical framework* involves the concepts, definitions, theory used in a given study, as well as references to the literature. It helps to determine the key variables that differ under certain circumstances [2, 231]. Sinclair [221] note that a *theoretical framework* can be understood as a planning of what it wants to do and what it wants to achieve. She adapted six questions initially proposed by [28] whose answers are conducive to outlining the plan. The questions are *"what do I know about the phenomenon that I want*

to study?", *"what types of knowledge are available to me (empirical,non-empirical, tacit, intuitive, moral or ethical)?"*, *"what theory will best guide my midwifery practice?"*, *"is this theory proven through theory-linked research?"*, *"what other theories are relevant to this practice?"*, and *"how can I apply these theories and findings in practice"*.

- *Knowledge management framework* include knowledge management activities comprised within the life cycle. KM frameworks can in turn be classified as:
  - *Prescriptive frameworks* provide guidelines for the types of procedures in knowledge management without providing details of how to achieve those procedures. These are the most widespread in the literature.
  - *Descriptive framework* identify attributes of knowledge management used within its description initiatives.
  - *"A combination of both, prescriptive and descriptive frameworks"*

Argent et al. [12] also mention the *modelling framework* which encompasses modelling, reusable libraries, tools for data manipulation, analysis, and development tasks, specifically those related to transformations, model coding, and visualisation. The *modelling frameworks* include features, architecture-related requirements, protocols, operating methods [12].

Greenfield et al. note that when very little code is required to be generated to complete the variability points in a domain-specific framework, we would be talking about a *completion framework* [94]. A software framework can address a *"well-defined, narrow problem domain, and using the abstractions in a model to define how the variability points in the framework must be filled"* [94]. The authors also emphasise that application developers will not create general-purpose code but will build customised variants of existing products that meet unique requirements, from small pieces of code in specific languages, all in order to complete frameworks [94]. Thereby, domain experts could encapsulate their knowledge and reuse it [94].

Moreover, frameworks also differentiate between *white-box* and *black-box* frameworks [122, 77]:

- In *white-box* frameworks, its implementation has to be understood in order to add the methods to the subclasses according to the superclass. *"White box"* frameworks inherit base classes from the framework and override predefined methods using patterns (e.g., templates). Here, the instances state is available for all methods.
- A *black-box* framework provides a set of components and users need only understand the external interface. *"Black box"* frameworks define interfaces including components (i.e.,

plugging them) via object composition. Here, information must be transmitted explicitly. They use less inheritance than *white-boxes*, and are based more on composition and delegation. In addition, they help the system to be better understood and they are also recommended when the system needs to evolve. A framework ideally tends to be a *black-box* of components that can be reused, providing the largest granularity, without knowing how they are implemented. These type of frameworks are more reusable, maintainable and they are easier to use but more difficult to extend because it is necessary to implement the interfaces and hooks in advance for a wide range of cases.

Shan et al. [215] differentiate between seven types of frameworks the information system:

- The *conceptual framework* referring to the overall architectural model.
- *Application framework* referring to the *"skeletal structure for an application solution"*.
- *Domain framework* referring to business sectors.
- *Platform framework* includes *"programming model and execution environment"*.
- *Component framework* referring to *"building blocks"*.
- *Service framework* referring to *"technical services model"*.
- *Development framework* referring to the development of a tool.

Within the framework of the platforms, it is worth mentioning web application frameworks. A Web Application Framework (WAF) tends to be a modular a reusable platform. It can be specialised through services or components, and which usually operates with Http in the browser. The frameworks often accept standards and technologies that speed up the development without a steep learning curve. WAFs can be further subdivided into *request-based* (e.g., WebWork, Struts, Beehive, etc.), *component-based* (e.g., JSF, Tapestry, Wicket, etc.), *hybrid* (e.g., RIFE), *meta* (e.g., Keel, Spring), *RIA-based* (e.g., DWR, Echo2, JSON-RPC-Java) [215]. Frameworks could be differentiated by whether they operate on the *front-end* side or on the *back-end* side [126]. Some examples of the most popular WAF in 2021 are Django, Angular, or Laravel, inter alia.

The Mobile Applications Development Framework (MADF) [242] supports the development of mobile applications that need to operate efficiently, enabling collaboration between users, improving their responsiveness and competitiveness, and offering context-aware and personalised responses to users. Added to this, there are numerous challenges in mobile application development due to the fact that platforms tend to fragment rather than join [123]. It

is therefore common for applications to have to be developed on each platform [123]. Some examples of the most popular MADF in 2021 are Swiftic, React Native, Xamarin, or Ionic, inter alia.

Frameworks are also related with Knowledge Management (KM) [224]. KM software tools are integrated within processes [224]. KM frameworks should be prescriptive, descriptive, and consistent. Moreover, their goals and strategies must be linked to the KM [224]. A KM framework can involve mainly five phases: 1) *"strategy"* includes organisation of the strategy itself and objectives; 2) *"evaluation"* of the current state of knowledge, scoping, prioritisation and technology solution; 3) *"development"* of block for the prioritisation of KM initiatives, and outline a plan; 4) *"validation"* includes pilot launch, review and update, and maintenance of knowledge processes; and 5) *"implementation"* covers the publication of the KM base and communication processes, as well as change management, maintenance and support processes for the creation of the KM, and continuous improvement [224].

A methodology *"is defined as a set of procedures that can be followed for achieving an objective and is more specific than a framework"* [224]. A framework *"provides guidelines as to how to carry out the procedure in such a way that it is consistent with a particular framework"* [224]. McMeekin et al. defines a methodological framework, as a *"structured guide to completing a process or procedure"* in a little more detail, they define it as a *"structured practical guidance or a tool to guide the user through a process, using stages or a step-by-step approach"* [152]. The IGI Global also defines it as *"an approach for making explicit and structuring how a given task is performed"* [90]. It should be noted that a methodology *"is defined as the group of methods used in a specified field"* [152].

The methodological frameworks can involve several aspects: *"a body of methods, rules and postulates employed by a particular procedure or set of procedures"*, *"a set of structured principles"*, *"an approach for structuring how a given task is performed"*, and *"a sequence of methods"* [152]. A methodological framework can also help standardise approaches, increase the robustness and confidence of particular findings, group common themes and metrics [152]. McMeekin et al. divide their development, mainly into five phases: 1) *"identifying the research question"*; 2) *"identifying relevant studies"*; 3) *"study selection"*; 4) *"charting the data"*; and 5) *"collating, summarising and reporting the results"*.

Patterns, frameworks, models, tools applied in a systematic way facilitate development, minimise cost and improve software quality [94].

2.5 Middlewares

It was in the NATO Software Engineering Conference in 1968 where the term middleware was first introduced within the Inverted Pyramid proposed by d'Agapeyeff (Figure 2.17) [225].

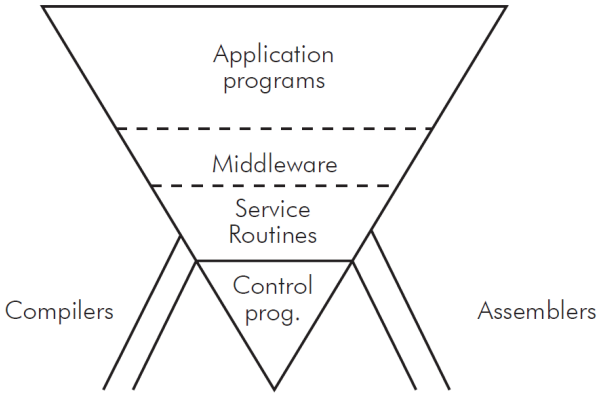


Fig. 2.17 *Inverted Pyramid proposed by d'Agapeyeff [225].*

The Oxford dictionary defines a middleware as *"a layer of software in a computer between the operating system and applications that provides additional facilities not provided by the operating system"* [68]. There are different definitions in the literature and generally, most authors agree on the same aspects. Ibrahim et al. [113] defines a middleware as an *"enabling technology for the deployment, execution and integration of applications"* providing software layers *"between the operating systems and applications"*. Bishop et al. defines a middleware, essentially as a software that connects two or more software or an interconnecting application for interaction/communication with others (e.g., *"applications, networks, hardware or operating systems"*) [35]. Bakken [19] points out that middlewares help to manage complexity and heterogeneity. Middleware frameworks provide solutions to heterogeneity to software programmers, *"location, concurrency, replication, failures and mobility"*. Bandyopadhyay et al. [26] summarises that middlewares are useful when it is difficult to establish a common standard because of the heterogeneity of devices, that they act as a link between components, or as an abstraction layer between applications of different domains, and that they provide APIs which abstract multiple details for the physical and service layers. A mobile middleware *"handles the dispersed services in a mobile environment and also provides the glue to varying types of devices"* [242]. It also bring together parts of a mobile software application [242]. Finally, in Figure 2.18, it can be seen visually where the middleware layer is located.

Bishop et al. propose a taxonomy for middleware which is subdivided into:

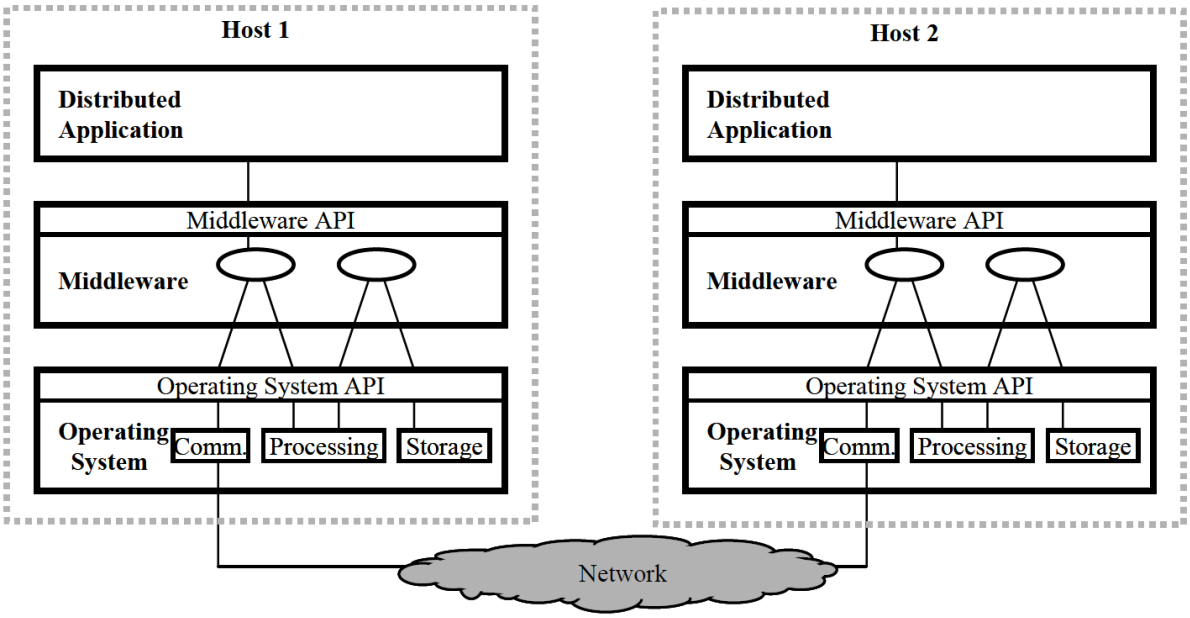


Fig. 2.18 *Middleware Layer in Context proposed by Bakken [19].*

- *Integration* referring to those middlewares that can be integrated in different ways in a heterogeneous environment, that can have different communication protocols and operating modes. This subcategory contains:
  - *Procedure-oriented middleware* where clients transform the parameters of a procedure into a message and server reverse it, and finally, it is processed in the application.
  - *Object Oriented Middleware* supports the communication between objects grouping several similar requests within a transaction. It may include a broker that acts as an intermediary between the sources and the servers.
  - *Message Oriented Middleware (MOM)* is further divided into *Passing/Queuing* and *Message Publish/Subscribe*. For the *Passing/Queuing*, the MON server acts as an intermediary by picking up the message from the queue (in a pre-determined order). For the *Message Publish/Subscribe*, the MON is a *"event-driven process"* which integrates a bus where listeners can be registered to received the notices sent by publishers. Subscribers also can request data through the bus.
  - *Component Based or Reflective Middleware* where components are selected *"either at build-time or at run-time"*. This middleware has a library of components to fulfil the needs of users, covering QoS requirements, and where changes and reconfigurations can be performed at runtime.

  - *Agents* can be considered as a middleware based on the definition itself of agents and their autonomy. This is because agents are entities that can communicate with each other. There are rules for this communication.

- *Applications* referring to those that fit into an application type such as *Data Access Middleware (DAM)*, *Desktop Middleware*, *Web-based Middleware*, *Real Time Middleware* and *Specialty Middleware*.

Middlewares (used in IoT environments) include several functional components such as [26]:

- *Interoperation* for sharing information by different communication networks (i.e., protocols). It also includes rules definition for the understanding of the information and the creation of models.

- *Context detection* allows identify an entity (e.g., person, object, interaction, etc.) status. In particular, a distinction is made here between three related types of context:

  - *Context detection* is referred to as relevant aspect from data collection.
  - *Context processing* is referred to as context data extraction, its processing and decision making.
  - *Context awareness* is based on both, *context detection* and *context awareness*.

- *Device discovery and management* for detecting neighbours in the network, semantic mapping of physical devices with semantics ones, and retrieving information.

- *Security and privacy* are responsible for confidentiality, authenticity and non-repudiation.

- *Managing data volume* of the network devices. Data can be of different nature (e.g., positional, environmental, historical, etc.).

Some examples of middlewares can be the following. On the one hand, HYDRA, UBIROAD, SOCRADES, and SYRENA include device management, interoperation, platform portability, security and privacy and HYDRA and UBIROAD incorporate context-awareness [26]. On the other hand, MYSIM, PERSE, SeGSeC, SeSCo, Broker, and WebDG are middlewares based on service composition that like the previous ones try to satisfy some or all of the properties of interoperability, discovery, adaptability, context awareness and QoS management [113].

The functional components of the core of the middleware are *interface protocols*, *device abstraction* ,*central control, context detection and management* and *application abstarction*

[25]. Blair et al. [36] already argued that traditional platforms such as Common Object Request Broker Architecture (CORBA) [95] and Distributed Component Object Model (DCOM) [236] are not flexible.

A middleware can be programmed to provide a library of functions, or as an external interface to a component that fits an Interface Definition Language and is used by the programmer to do code, or another form of middleware can support a native distribution (e.g., Remote Method Invocation) [19].

2.6 Development and Operations (DevOps)

From the combination of terms, development and operations comes DevOps. DevOps *"is a set of practices intended to reduce the time between committing a change to a system and the change being placed into normal production, while ensuring high quality."* [30]. It also means *"end-to-end automation in software development and delivery"* [71]. Specifically refers to *"automated development, deployment and infrastructure monitoring"* [71]. Figure 2.19 shows the life cycle of DevOps processes.

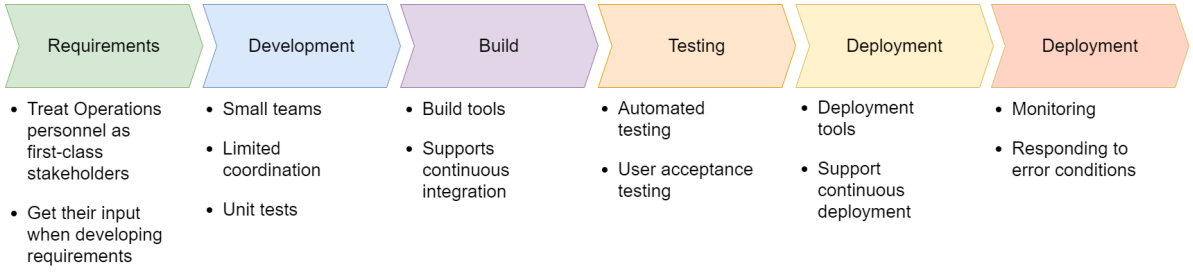


Fig. 2.19 *Life cycle of DevOps proposed by Bass et al. [30].*

Build phase comprises a set of tools to *"software development and service life cycle"* which include *"compiling code, managing dependencies, generating documentation, running tests, or deploying an application to different environments"* [71]. Some examples of tools for building applications can be Apache Ant, Maven; for continuous-integration tools it can be mention Jenkins, TeamCity, Bamboo, Puppet or Chef; Loggy or Graylog for debugging; and as monitoring system (i.e., CPU, RAM, network traffic, etc.) tools, New relic or Cacti [71]. DevOps provides a framework *"for developing, deploying, and managing the microservices container ecosystem"* [71].

2.7 Cloud Computing

Services are also related to *cloud computing* but also to *"the hardware and software systems that provides those services"* [13]. *"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."* [154]. Cloud computing can be also defined as an style of computing paradigm which is based on the concept of resource virtualization, so that when a system needs more resources, these can be provided on demand and they can also be unassigned whether does not need them [217]. This alleviates the overload, favours QoS guarantees, and provides scalability and flexibility but some of the inconvenient remains (e.g., a continuous internet connection is required, data security, its viability depends entirely on the transport of data, etc.) [217].

Cloud computing is composed by *"five essential characteristics"* that are *"on-demand self-service"*, *"broad network access"*, *"resource pooling"*, *"rapid elasticity"*, and *"measured service"* [154]. Furthermore, Cloud computing presents three models [154, 182, 118]:

- *Software as a service (SaaS)* model provides software on-demand through a user interface, specifically, Applications Service Provider (ASP) introduces the SaaS *to deliver complex business processes and transactions as a service, while permitting applications be constructed on the fly and services to be reused everywhere and by anybody* [182].

- *Platform as a service (PaaS)* model supplies technologies such as virtual serves, operating systems, developer tools, storage, inter-alia but the hardware and software resources are managed by the provider.

- *Infrastructure as a service (IaaS)* model provide software and hardware resources that are managed by the clients, i.e., developers can install, for example, operating systems, manage the database, etc.

Data processing tasks in the cloud have been shown to be efficient because of the computing capacity that can be provided. However, the volume of data generated and its transmission causes bottlenecks which has an impact when real-time processing is needed to make decisions [219], or for *"latency-sensitive applications, which require nodes in the vicinity to meet their delay requirements"* [39].

Three paradigms closely related to cloud computing should be highlighted [267]:

- *Edge computing* arises to collect and process data in a device is to bring data computation (e.g., initial filtering of data, elimination of redundant data, removal of noise, etc.) closer to the source nodes, at the edge of the network (gateway), in order to minimising the amount of data to be transmitted and providing solutions to concerns about response time in the requirements and quality properties in terms of remaining battery, bandwidth, data security, etc. Edge devices are the path between data sources and data centers in the cloud where services and data can also be requested from the cloud and task can be performed in the cloud [219].

- *Fog computing "is a highly virtualized platform that provides compute, storage, and networking services betweenend devices and traditional Cloud Computing Data Centers, typically, but not exclusively located at the edge of network."* [39]. Among its features are *"low latency and location awareness, widespread geographical distribution, mobility, very large number of nodes, predominant role of wireless access, strong presence of streaming and real time applications, heterogeneity"* [39].

  Some authors refer to fog computing and edge computing as similar paradigms with the differentiating nuance that *edge computing* is about things and *fog computing* is about infrastructure [219].

- *Mobile cloud computing* follows *"the trend to extend the cloud to the edge of networks"* alleviating problems arising from resource constraints, frequent disconnections and mobility [110]. In this regards, *"portable devices sense and learn the status of devices and the context related to their mobility and networking in order to better support mobile applications in an ad hoc communication environment"*. Providers external to the mobile device act as a host to run applications, store data or perform processing [79]. However, there are several points of view to define this concept which can be summarised as follows [79]:

  - Running applications on a more powerful resource provider external to the mobile device, e.g., view Gmail mail from the device that is remotely connected to an external server.
  - To consider other mobile devices, which are neighbours, providers of the service for which a peer-to-peer connection is established.
  - Mobile devices relegate their work to other devices connected to each other (local cloud) and in turn connected to remote servers (cloud).

This is also where *mobicloud applications* come in. *Mobicloud computing* serves as a connector between the information sources of the cloud and mobile computing service, and acts as a knowledge hub [110].

2.8 Devices and Networks

This section includes information about wearables to collect physiological data and sensors to collect context data. It also covers the evolution of sensor networks, nodes categorisation, and communication and propagation techniques used in such networks.

2.8.1 Sensor Networks and Wearables

Sensors and wearables are key pieces to monitor *context* and to provide *context-aware* applications [66, 194]. Monitoring is, among others, a field of telemedicine (also known as *connected health*) where more emphasis is placed on the connection between the medical team and patients. Users carrying their mobile device in their pocket or nearby allows the device itself to act as a storage service or as a gateway to access data [37]. Wireless technologies, microcontrollers, together with other microtechnologies (e.g. individual chips) have enabled wireless connectivity between individual sensors. Sensors are used for collect data from the environment [66, 194] but whether devices are in or around the body itself, and whether there is an interconnection between them give rise to wireless Body Area Network (BAN) networks, Wireless Body Area Networks (WBANs) or which is also known as Body Sensor Network (BSN) [235]. Wireless Body Area Network (WBAN) was first coined by Van Dam et al. in 2001 in order to exploit resources (wireless technologies) within the field of telemedicine. WBAN involves small intelligent devices attached to the body that provide continuous monitoring and wireless communicacion connections [246]. Figure 2.20 shows an example of what a WBAN could look like [138].

Moreover, Hybrid BAN concept (h-BAN) arises from the possibility of interconnection of several sensors and additional communication with the outside world [235]. These microcontrollers and sensors open up new opportunities for applications using wireless networks (e.g., Wireless Intelligent SEnsor (WISE) prototype for data collection and filtering) [124].

Related to sensors, it is worth mentioning the wearables. The development of wearables was prompted by the possibility of continuous monitoring at home or community settings and the benefits this brings [37]. Wearable concept was born at MIT Media Lab's via the Wearable Computing Project. Steven Mann created his first wearable in 1970, and then transferred his

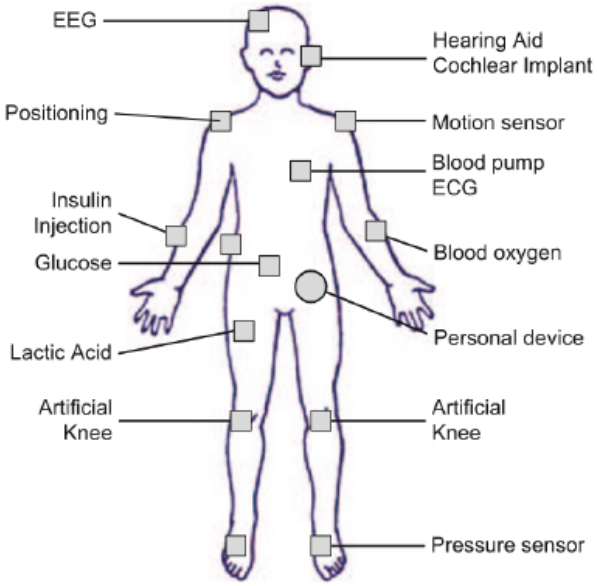


Fig. 2.20 *Wireless Body Area Network proposed by Latré et al. [138].*

invention to MIT in 1991. Another prominent researcher, a colleague of Mann's, has been Tahd Starner. The two share an interest in research into these devices, and have developed vision systems based on wearable glasses [254] Wearables tend to be hands-free use, and they should be proactive and collect data without the explicit intervention of the user [66, 194]. Wearables are characterised by their small size and their ability to collect various types of physiological data in a continuous (i.e., 24x7) and non-intrusive manner. This is intended to improve the user's quality of life. Over the years, efforts have been made to improve communications, security, and to reduce the power consumption of these devices, the latter being one of the main handicaps. They come in various forms. Since its origin, different variants with different formats have emerged. In recent years, designs for watches, bracelets, wristbands, glasses, jewellery, wearables, skin patches, among others, have become predominant. [212]. MediWatch [172] was the first wristband designed with miniature sensors for blood pressure monitoring.

Seneviratne et al. [212] group wearable devices into three categories (Figure 2.21):

- Accessories comprise devices that are worn by users but that are not considered to be a main part of the garment (e.g., smart watches, smart belts, smart glasses, etc.).
- E-Textiles refers to the group of devices that use conductive fabrics and embedding sensors as materials (e.g., foot or hand worn devices, smart garments, etc.).
- E-Patches are those that can be attached or tattooed onto the skin, which are actually tiny embedded sensors.

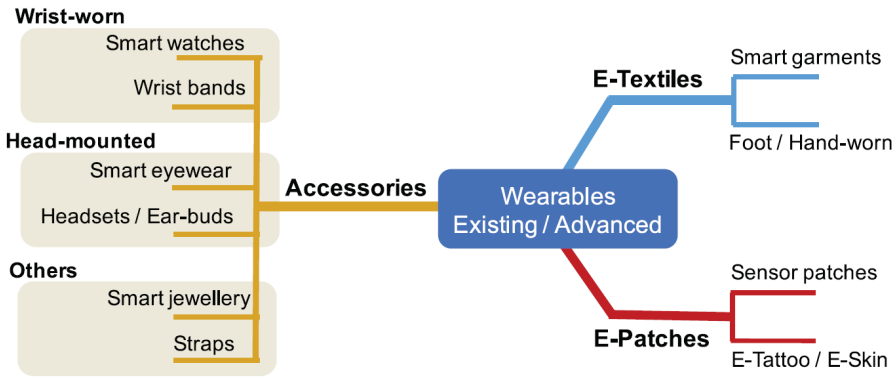


Fig. 2.21 *Wearable devices classification proposed by Seneviratne et al. [212].*

Research has also focused on provide hybrid solutions that combine wireless and e-textile (also known as smart textile) technology [37]. Smart textile use materials and nanotechnology, as proposed by researchers of Georgia Institute of Technology who patented the Smartshirt which collects analogue signals by means of conductive fibre sensors and transfers them through conductive fibres [235]. These technologies drove an increase in the number of sensors, biosensors implanted in the body and other devices placed in different parts of the user's body for long-term and continuous data collection [235].

WSNs are present in situations or scenarios where requirements cannot be fully predicted or where systems have to adapt to changing conditions. Therefore, *"the ability to reconfigure portions of the software running on WSN nodes become imperative"* [159]. It is also common to deploy heterogeneous sensors (i.e., with different characteristics), and therefore, the need to deploy codes adapted to such characteristics [159]. WSN systems require to be configured *on-the-fly* and *on-demand* by mobile elements [159]. In addition, it is crucial to be very precise on exactly what functionality needs to be upgraded (*fine-grained*) to minimise energy consumption [159].

Devices Sample

During the first period of the thesis, different devices and their characteristics were studied. Here, we summarise some of the available alternatives that were assessed for incorporation in the application domains subject of study.

One of the most interesting wearables found was the E4 wristband for real-time physiological data streaming and visualisation proposed by empatica [73] (Figure 2.22). This wristband incorporates four devices: a photoplethysmography for continuous heart rate, a 3-axis ac-

celerometer, a body thermometer, and an electrodermal activity to measure skin conductance. This device incorporate a flash memory to store the data series for each session.

Fig. 2.22 *E4 wristband for real-time physiological data streaming and visualisation proposed by empatica [73].*

The following should also be mentioned for the measurement of physiological measurements (Figure 2.23):

- *Embrace* [74] was designed to monitor epilepsy. It has an advanced electrodermal activity sensor with electrodes, gyroscope, 3-axis accelerometer and motion detection, thermometer, built-in DSP. In addition, it has a small vibration motor and tactile alarm for personal notifications, and an event-dial button (with tactile feedback).
- *Neüma* [169] was designed to monitor stress. It includes sensors for measuring conductance, ambient temperature and movement (3-axis accelerometer).
- *OxiPatch Oxirate* [177] is a wearable to monitor heart rate, heart rate variability, blood oxygen saturation, respiration rate, and physical activity (actigraphy).
- *iWatch* [11] includes accelerometer, gyroscope, heart rate sensor, nanometric altimeter and blood oxygen sensor.

- *Microsoft Band* [156] designed to detect calories burned, number of steps or distance covered, and sleep monitoring, as well as height/stair detection sensor and pedometer. It has bluetooth 4.0 LE connection.
- *Pip* [162] is a biosensor device held between the thumb and forefinger. It measures sweat and electrodermal activity associated with stress levels. The device, which connects via Bluetooth with smart devices.
- *Gear 2 Neo* [203] integrates pulse, accelerometer and gyroscope sensors.
- *Muse* [163] is a stress detector headband, which measures brain activity from electrical signals monitored by seven sensors.
- *Q-sensor* [115] wristband that records physiological signals of stress and arousal through slight electrical changes of the skin. It is a wristband that allows people's stress to be tracked during daily activities. It is being used in children with autism.
- *Polar H7-9 Heart Rate Sensor* [187] is a chest strap to monitor heart rate. Connection via Bluetooth.

One of the most interesting devices found to monitor environmental conditions was the Samsung Galaxy S4 (Figure 2.24) because it incorporates nine sensors in a single device, including temperature and humidity sensors [171].

The following should also be mentioned for the measurement of environmental conditions (Figure 2.25):

- *Sensor Tag* [116] includes temperature, humidity, light sensor, accelerometer, magnetometer, gyroscope and pressure sensors. It uses Bluetooth low energy and connects to iOS and Android devices.
- *Sensordrone* [213] includes ambient temperature, humidity and pressure, non-contact infrared temperature, oxidising and reducing gases. Moreover, it also integrates for measuring carbon monoxide levels and hydrogen sulphide levels. The device connects to iOS and Android devices via Bluetooth Low Energy.
- *Spotter-Uniq* [261] includes temperature, humidity, and sound sensors. The device connects to iOS and Android devices, and the data transmission is via JSON.
- *Spotter-Quirky* [261] includes temperature, humidity, and sound sensors, similar to the previous one and from the same company. Displays notifications.

(a) *Embrace proposed by empatica [74].*

(b) *Neüma proposed by Neumitra [169].*

(c) *OxiPatch Oxirate proposed by OxiPatch [177].*

(d) *iWatch proposed by Apple [11]*

(e) *Microsoft Band proposed by Microsoft [156].*

(f) *Pip proposed by Niall Murray [162].*

(g) *Gear 2 Neo proposed by Samsung [203].*

(h) *Muse proposed by Muse [163].*

(i) *Q-sensor proposed by the Massachusetts Institute of Technology (MIT) [115].*

(j) *Polar H7-10 proposed by Polar [187].*

Fig. 2.23 *Sample of devices studied for physiological measurements.*

- *Sensor Cube* [61] include sensors for light, pressure, air quality, temperature, humidity, and noise. The data transmission is via JSON.
- *Blue Radios* [191] includes temperature and light sensors, and accelerometer. Connection is via Bluetooth.

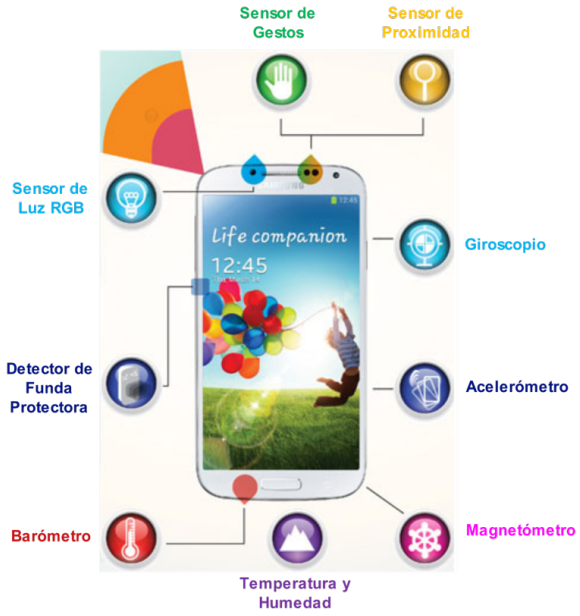


Fig. 2.24 *Mobile device that stands out because it integrates environmental temperature and humidity sensors in addition to the conventional ones [171].*

(a) *Sensor Tag proposed by Texas Instruments [116].*

(b) *Sensordrone proposed by Sensorcon [213].*

(c) *Spotter-Uniq proposed by Wink [261].*

(d) *Spotter-Quirky proposed by Wink [261].*

(e) *Cube Sensors proposed by CubeSensors [61].*

BlueRadios

(f) *Blue Radios proposed by Blue Radios [191].*

Fig. 2.25 *Sample of devices studied for environmental measurements.*

2.8.2 Nodes Categorisation

Sensor networks are a significant improvement over traditional sensors. Sensor networks involve the deployment of a large number of sensors and their topology is frequently subject to

changes. These sensors can sense data (also known as *sources*), process data and send data to *base stations (BS)* (also known as *sinks*) [223].

Sensors capabilities are limited (i.e., power, computation, storage) but sufficient to transmit raw data to central nodes or even to perform computational operations on them to send processed data [4].

Regarding sensors deployment, it can be predetermined or random but the latter implies that nodes possess self-organising capabilities. The density of nodes in the deployment favours the use of multihop communications rather than single hop communications. Multihop communications can be expected to consume less than single hop communications but signal propagation effects may occur [4]. In general, there are several factors that influence the design of sensor networks mainly referred as *"fault tolerance, scalability, production costs, operating environment, sensor network topology, hardware constraints, transmission media, and power consumption"* [4].

The inclusion of Mobile Entities or Elements (MEs) which are also called Mobile Ubiquitous LAN Extensions (MULEs) within a wireless sensor network allows these elements to collect sensor data, store it temporarily (*buffer it*) and deliver it to wired access points [214]. Shah et al. [214] point out several benefits that a mobile infrastructure could provide, e.g., minimising network infrastructure (a fixed base station), increasing robustness in the network, and prevent sensor nodes from supporting multihop connections. MEs can be used to improve the performance of the network (e.g., energy efficiency, connectivity, reliability, end-to-end delays, etc.) [223]. In particular, it could alleviate the hotspot problems that occurs when multihop connections exist and affects the energy consumption of nodes close to the BSs which is higher because they have more traffic [223]. If one part of the network is affected by a power problem, this has repercussions for the rest of the network [223]. MEs are also data carriers and a tool for extending network lifetime. The MEs [223] follow a trajectory classified as:

- *random* with movements in any direction;
- *controlled* because the movement is controlled by the users; and
- *uncontrolled* because the trajectory is fixed and predefined.

Finally, the network components and data collection can be classified into three groups [223]:

- *Mobile base station (MBS)* is the combination of mobility and traditional base stations. It can be more than one MBS deployed.

- *Mobile data carrier* refers when the BS is static and there are one or more transmission nodes.
- *Normal sensor nodes* are regular nodes to sense variables.

2.8.3 Communication/Propagation Techniques

In WSN *"of small nodes with sensing, computation, and wireless communications capabilities"* several issues have been subject of study, such as routing and data dissemination protocols, energy consumption, flexibility, low cost, fault tolerance, speed deployment, scalability, or topology changes [86].

Communications is the most energy-intensive part of a network, and there are different approaches to mitigate this problem [86]:

- Node-centric communications is the paradigm used in Internet.
- Data-centric is based in top-to-bottom solutions and it is supported by hierarchical and peer-to-peer topologies.
- Position-centric uses any node for forwarding.

In parallel, WSN network topology is usually unknown or change over time, because several reasons such as the existence of mobile nodes or because nodes are added or removed. However, despite this limitation, it is often necessary to transmit an update in the code or in the configuration of the nodes of the network.

Flooding Technique

The simplest way to approach this problem is through the *flooding technique* [148], where, when a node receives a message, it retransmits it to all its neighbouring nodes, which reproduce this behaviour. In this way, the information will reach all nodes in the network at least once. However, this technique produces what is known as a broadcast storm: the same message will reach the same node several times through different routes, overloading the network with unnecessary communications. In addition, for the correct functioning of this technique, it is necessary that nodes know how to identify redundant messages, thereby avoiding a circular retransmission of the message. There are some modifications based on this basic technique, such as the one proposed in [272], where the diffusion of the message is limited according to the distance in hops from the sink node.

Flooded broadcasting is common in WSN, but power is expensive and redundancy, contention and collision are also frequent. Data gathering process must be efficient to guarantee the lifetime of the network. One message is required to build the topology of the network. In the topology generation process, when a node is deployed, it will be broadcast (i.e., sent by flood) a discovery message, and receivers will rebroadcast. These messages contain node information (e.g., node ID, level, etc.). Each node can receive several messages which is useful for it to know who its neighbours are. The information contained within the received message is used by each node to select its parent. There are several strategies to perform the election [273]:

- Earliest-first, the parent will be the sender of the first message received.
- Randomized, the parent will be chosen randomly from neighbouring nodes, without any priority.
- Nearest-first, the nearest neighbour will be chosen as a parent.
- Weighted-randomized, the nodes assign a score to their possible parents based on the number of neighbours of the latter.

Spanning Tree Communication

More advanced techniques involve the construction of a *Spanning Tree*, as a hierarchical structure of communication from the network nodes to the sink or Gateway node. Thus, redundant communications and message implosion can be avoided. In this approach, each node selects a single parent node, through which it will send and receive information. However, the construction of this type of structure is costly, as it is usually done using flooding or similar techniques. Because of this, it is not a suitable technique for networks where there is mobility or high dynamicity, as it would be necessary to continuously build or adapt the tree. Further, the selection of the parent node is crucial for the optimisation of future communications, and it is desirable that the spanning tree is kept to a minimum to reduce the cost of transmission. There are several distributed techniques for the construction of Spanning Trees. For example, in [67] an approach based on the Gallager-Humblet-Spira algorithm [84] is proposed. Each node starts the construction of the tree with its neighbours, in a concurrent and distributed way. When two tree fragments meet, they are joined into a single fragment through the edge with the lowest weight that connect the two fragments. In the case of the proposal, for WSNs, the lowest weight is represented as the edge with best bandwidth.

Gossip-based Protocols

Gossip-based protocols can also be found [34]. These emerge as a probabilistic propagation technique based on epidemic replication, aiming at decentralised operation and high scalability. Unlike flooding techniques, a node does not transmit the message to all its neighbours, but to one (or several) selected randomly. This node, in turn, will select another (or others) to which it will retransmit the information. In this way, duplicity in message transmission is eliminated or considerably reduced, achieving high scalability and avoiding the problem of broadcast storm. One of the main problems of this protocol is the intervention of malicious nodes, which can modify the content of the message. In [9] a modification to the basic gossip approach is proposed. Instead of electing randomly a neighbour node to send the message, it elects a neighbour node based on the distance to it and the distance from it to the sink node. It proposes two approaches, one using the euclidean distance and another using the city block distance.

Clustering-based Techniques

Finally, *clustering-based techniques* can be found, where nodes are grouped together and elect a leader or cluster head. In this way, communication between groups only takes place through the cluster heads, reducing inter-group communication and providing a hierarchical communication structure. In this type of technique, there are two fundamental steps: the creation of the cluster and the election of the cluster head. A correct formation of clusters leads to a reduction of communication between distant nodes. Moreover, the cluster head can easily become a bottleneck, as it will be the gateway of its cluster. As a result, the resources of the cluster head (e.g., battery) will be consumed at a higher rate than for the other nodes. Among this type of techniques, we can find the Low-Energy Adaptive Clustering Hierarchy ("LEACH") [101], which is a self-adaptive and self-organised protocol. The LEACH configuration is divided into rounds, in each round a new cluster head in each cluster is elected based on a probability. The elected node cannot be elected again in the next rounds until all the nodes in the cluster have played this role. In this way, the workload is distributed among the nodes in the network. One of the main problems with this approach is that it is not designed for mobile networks. This is because disconnecting nodes from their cluster head would require reconfiguring the clusters to maintain a hierarchical communication, thus, high mobility would require constant reconfiguration.

2.9 Network Simulators

Simulators are tools that allow the emulation of real situations in controlled environments. Simulators offer us the possibility of establishing configurations with a high level of specification and precision, which allows us to create scenarios with characteristics very similar to real scenarios. In turn, this means that the results obtained during the tests performed should be close to those that would be obtained in a real scenario.

In this section, it can be found a selection and summary of each of the simulators that would be most suitable for the context of use in this proposal.

TOSSIM

TOSSIM [239] is an application simulator for TinyOS. It provides an interface to be used by C++ or Python clients. It bases its simulation on the occurrence of discrete events and allows different levels of simulation, such as hardware interruptions or the reception of network packets. It can simulate mobility but does not simulate power consumption. It is a simple to use simulator, however the code developed in the simulator may not be compatible with real TinyOS motes, as the simulator makes different simplifying assumptions.

J-Sim

J-Sim [117] is a simulation environment based on the Autonomous Component Architecture (ACA) and written in Java. Although it was not initially designed to simulate networks or sensor networks, it was extended to include different functionalities to represent sensor and sink nodes, as well as wireless propagation models [226]. One of its main advantages is that, by using Java, it is platform independent. Moreover, it is scalable and can simulate a large number of nodes compared to other simulators. However, the design of the framework makes it difficult for users to add new protocols, making its functionality outdated over time [57].

NetSim

NetSim [168] is a network simulator for Cisco Systems, programmed in C and based on an object-oriented approach. It is frequently used as a training and testing tool for network technicians in corporate networks. Although it provides an intuitive and user-friendly interface, it is proprietary software.

OPNET

OPNET (Optimized Network Engineering Tool) [175] is a commercial simulator written in C and C++, which allows simulate several heterogeneous networks. It was initially developed to simulate fixed networks, but it has been extended to simulate wireless networks as well, although there is a lack of recent wireless protocols.

OPNET++

OMNeT++ (Objective Modular Network Testbed in C++) [174] is a non-commercial discrete simulator. It provides a powerful Graphical Interface and the INET framework allows to simulate mobile networks. Although it provides a large catalogue of protocols, it can be considered limited compared to other simulators such as ns-3. It has a commercial version entitled OMNEST.

GloMoSim

GloMoSim (Global Mobile Information System Simulator) [17] is a simulator devised to use the parallel capability offered by Parsec, a C-based simulation language. Although it was devised to support wireless networks, currently, the project is no longer under active development.

QualNet

QualNet [234] is the commercial version of GloMoSim. It is designed to perform high-speed and large-scale simulations of a wide variety of networks (commercial, military and governmental). It also offers a catalogue of simulation models of commercial devices.

Simulink

Simulink is a MATLAB extension that provides visual programming for model simulation. It can be used to simulate sensor networks and mobile networks, however, it is based on mathematical models and although it can simulate different communication models, it does not simulate the full protocol stack, as other simulators do.

Avrora

Avrora [16] allows the simulation of AVR microcontrollers and network sensors. It was developed under a research project of the UCLA compiler group. Among its most notable

features is that it can simulate microcontroller programs at instruction level, instead using software models.

Ns-3

Ns-3 [173] is a network simulator implemented in C++. It is licensed under GNU GPL and is widely used in research and education. It is not compatible with its previous versions, ns-2 and ns-1, in order to overcome the problems they presented. Currently, ns-3 is being actively developed and maintained. Among its most notable features is that it incorporates APIs that allow the integration of real code, such as Berkeley sockets or POSIX threads. It incorporates numerous protocols, both for wired and wireless networks, and models, for example, to simulate battery consumption.

2.10 Summary

This chapter comprises a study of the different paradigms, approaches and technologies related to the predefined objectives of this thesis.

Current monitoring systems should incorporate adaptation and self-adaptation mechanisms from which adapted systems can be provided that can also adapt autonomously to changes in context. The context information can be classified into computing system context, user context, physical context and temporal context which allows to provide general and global information on the system. There are also several dimensions that bring together the different perspectives and aspects to consider when adapting SAS systems.

Application building is feasible based on the integration of several applications or composition of services. Services can be part of an aggregation or a composition, which also depends on orchestration and choreography. SOA and MSA support the decomposition of a system into services, although concerns, such as deployment, user interface, flexibility, management, scalability and service size present certain differences [51]. SOA was initially designed to operate in static environments but not straightforwardly in dynamic ones (e.g., IoT) [41].

Over the years, a number of architectural approaches have emerged. SOA, SOA 2.0, EDA, ROA, agent-based, DSSA and other specific proposals that have been taken as a reference by including mechanisms that support adaptation and self-adaptation. From the latter, it can be concluded that they include cyclical mechanisms in which several phases can be distinguished.

The frameworks include reuse techniques from different perspectives, libraries, design, code, parts of the system, etc. They provide a layer of abstraction that allow users to access their

functionality in a transparent way. The middlewares also provide an abstraction layer. However, these are at a lower level than the frameworks, specifically between the operating system and the application, albeit there are proposals that bring the two approaches closer together. Another emerging practice is DEvOps, which focuses on reducing the time for possible changes and deployment.

The cloud computing is a key concept in ubiquitous environments. The edge computing and mobile cloud computing paradigms aim to bring data collection and processing closer to the source nodes and to alleviate resource congestion problems respectively. With respect to devices, sensors and wearables collect physiological and context data, and depending on their capabilities support raw data transmission and optionally on-node processing (in-network processing). Communication and propagation techniques are necessary for this and for propagating possible modifications, code or other data through the network infrastructure.

Finally, simulators are used to emulate scenarios very close to real environments. This favours testing with different network configurations (e.g., with varying number of devices, range, categorisation, mobility types, etc.).

Chapter 3

Related Work

Chapter Abstract

This chapter presents a selection of relevant proposals related to the work developed in this thesis. These can be divided into three main pillars: (1) design and development of monitoring systems in order to increase reuse for different application domains; (2) autonomous deployment in the WSNs minimising human intervention, and supporting reconfiguration at runtime; and (3) data gathering in WSNs with mobile elements. A total of 30 papers have been summarised, 63.3% of which belong to scientific journals. Among the main conclusions that can be drawn from the review are that, generally, the proposals are ad-hoc for specific scenarios, making it difficult to adapt them to other monitoring scenarios different from those for which they were designed; they are designed for reliable communication networks or for networks in which the sensor nodes have a direct connection to the Internet; or they assume that the trajectory and behaviour of the mobile elements (i.e., data collectors) can be controlled.

Chapter Contents

3.1 Introduction

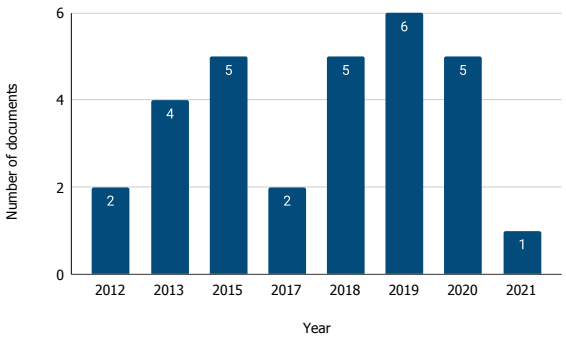
The proposals found in the literature aim to contribute with new design approaches that allow developers to focus on the specific elements of the application domain, abstracting from low-level details of the lower layers of the frameworks. In this way, two software programmer roles can be distinguished, the domain-specific programmer who uses the framework and its tools to design and configure specific applications for WSNs, and the system programmer, who is in charge of developing the functionality of the framework in a modular way and with a generic approach independent of the application domain (i.e., services, microservices, libraries, etc.) addressing and abstracting the heterogeneity of WSNs (e.g., communication protocols, platforms and languages). This thesis work presents a framework to address three basic pillars: (1) design and development of monitoring systems in order to increase reuse for different application domains (Subsection 3.2); (2) autonomous deployment in the WSNs minimising human intervention, and supporting reconfiguration at runtime (Subsection 3.3); and (3) data gathering in WSNs with mobile elements (Subsection 3.4).

A total of 30 proposals, published between 2012 and 2021 (Figure 3.1a), have been selected and analysed in this chapter as research works most directly related to the thesis work. Of these, 63.3% are journal articles (Figure 3.1b) and the most frequently used keywords include IoT, WSN, Sensor Nodes, Energy Efficiency, Mobile Elements or Data Fusion (Figure 3.1c).

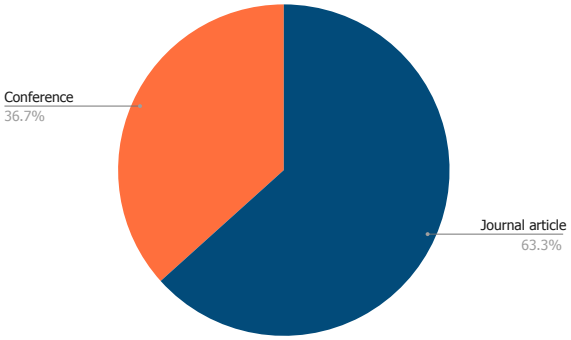
3.2 Framework for Monitoring Systems Design and Development

In [119], a visual editor is proposed to define virtual sensors using a JSON notation. The editor allows the design of basic operations, combining inputs from different sensors but more complex operations must be script-programmed. It supports the detection of complex events (e.g., detecting whether a person is seated) by combining different information and it incorporates a visual debugging mechanism.

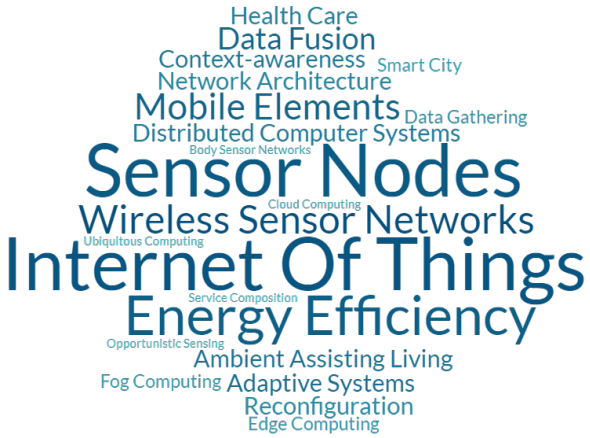
In [240], applications are modelled as Directed Acyclic Graphs where each node includes tasks represented by a colour. Monitoring tasks correspond to green circles, single data aggregation tasks correspond to red circles, and sink tasks correspond to yellow circles. Edges between circles define dependencies between tasks. The authors represent the application requirements as 2-tuple *(T,D)* where T corresponding with the set of tasks and D with the time of the application. Tasks are also represented as 4-tuple *(s, x, y, r)* where *s* is a service (e.g.,



(a) *Year of publication of the documents.*



(b) *Types of documents.*



(c) *Word cloud of the key words of the documents.*

Fig. 3.1 *Information of the most relevant proposals selected within the review.*

kind of sensor), x and y are coordinates of location, and r is communication rate to represent how often the aggregated data is sent to the parent nodes.

In [245], the Node-RED programming tool is extended with two new building blocks to interact with the IoT-Directory [38] and uniformly access the registered sensors and actuators in a smart city. The main objective is to facilitate city operators to develop smart city applications through a flow-based visual programming paradigm. The IoT-Directory [38] is an ontology-based approach to homogenise access to different devices (sensors/actuators) and abstract their physical particularities, in an effort to facilitate the interoperability of applications across IoT platforms.

In [204], the authors present a framework in the context of the Internet of Services (IoS). The proposal makes a division between adaptation logic and application logic. Its main objective is to allow the design of applications based on the composition of services, enabling the self-adaptation at runtime of the application through the substitution of services or their composition. In addition, it allows activities to be defined in an abstract way and to be set up at runtime according to the specific execution context.

In [82], it is proposed a framework to support the development and deployment of applications in Collaborative Body Sensor Networks (CBSNs), namely C-SPINE. Among the main features, there are components for In-Node Processing, which allow defining functions such as filters or aggregation operations in the sensor nodes. To demonstrate the usefulness of the framework, an application is developed to detect handshakes and the emotions of the users using them automatically, through different body sensors.

In [129], authors provide a SOA-based middleware to support WSN virtualisation. For this purpose, it deploys at source and sink nodes level a set of services that aggregate the data from the physical sensors to fix the behaviour of virtual sensors.

In [99], the authors present a framework to support the representation of a system from a statechart diagram, it provides libraries associated with the state diagram and the actions to be carried out in the system. It also includes a visual editor for the domain-specific programmer to design the system using a state diagram. The editor suggests actions to be included within the system and when the system programmer adds new actions the editor is automatically upgraded. The state diagram is represented as a JSON file and it is sent to the sensor nodes which will execute the specified actions. The actions considered are categorised as network, data processing, sensor service, and system actions. This framework is designed to operate on different platforms but is currently only available for StateOS.

Research in [186] presents a platform, namely REACT, designed to allow users to define adaptive behaviours at runtime. It is based in the definition of two models, the adaptation

options specification and the target system specification in order to provide structural and parametric adaptation. These models are defined by the domain-specific programmer using Clafer language [18] (adaptation options specification) and UML (target system specification). Through REACT, users can modify at runtime adaptation behaviour for overlay, P2P or underlay networks.

3.3 System Adaptation and Configuration Upgrade at Runtime in WSN

To facilitate the development and maintenance of monitoring applications in this context, some proposals are beginning to emerge that seek to establish a common platform, capable of adapting the specific needs of a particular monitoring scenario, yet providing all the common elements necessary to carry out the monitoring.

In [265], the authors propose a method for reconfiguring WSNs using mobile agents, called MAAR. These agents send code to nodes to modify the application requirements and adapt it to dynamic environments. The approach injects mobile agents to facilitate collaboration between sensors, controlling sensor state, context, environmental data, operations, sharing sample and code, and improves performance in terms of communication time and energy cost. In MAAR, the mobile agent can be equipped with a rules engine and also includes four components to perform conversion, sampling, a trigger for reconfiguration and code base. The reconfiguration process is divided into three phases: recognition, decision-making, and detection of new codes for the application (automatic detection if the rule engineering finds a match or manual if the administrator has to upload for the agent), and sending codes (i.e., the application codes to the sensor nodes). The evaluation has been performed on MicaZ motes that comprise light, temperature and smoke. Data flows through mobile agents to the sink responsible for integration. The reconfiguration request arises at the sensors due to environmental changes and is transmitted to the sink. The reconfiguration is processed by the mobile agents, not by the sensors. The communication time and energy cost are both measured.

Research in [192] presents an architecture based on fog computing to provide an intermediary layer between sensor nodes and cloud where to execute task and to improve connectivity. It also presents a prototype of a Smart e-Health Gateway called UT-GATE and an IoT-based case study, called Early Warning Score. It addresses issues of mobility, efficiency, scalability, reliability and explores where the gateway is best placed to provide high-level services (i.e., storage, data processing, data mining, etc.).

In [114], ActivFORMS is presented from which designers model and verify a feedback loop that is connected to external elements. ActivFORMS can be deployed in a runtime environment to perform the adaptation for the different quality goals. Usually, those are stochastic models to represent uncertainty by means of parameters. An example is illustrated using the DeltaIoT IoT application to address how to minimise the energy consumption of the motes while guaranteeing performance despite interference in communications or changes in the network. A simulation is carried out and the average time required for adaptation is measured.

Research in [55] addresses the problem of when, where, and which services to host on sensor nodes considering the computational, and storage limitations of the fog nodes. The authors propose a distributed algorithm, called adaptive for configuration (AFC), and a system model that can represent the energy consumption, service delay, and performance in terms of long term service delay and long term energy consumption. Performance evaluation is also carried out by simulations.

Research in [99] includes an Over The Air (OTA) module integrated for downloading software applications and upgrades. The downloaded application starts its propagation in the WSN sink (gateway).

In [264], the authors present an architecture, called BSIS, composed of three layers (edge, network, and cloud) that supports hybrid edge-cloud IoT base station system. Edge nodes collect data from sensors and devices, and there is a flexible layer in which the nodes are manually or automatically reconfigured in the cloud and tasks are also executed in real time.

Research in [142] proposes a Continuous Data-Flow (CDF) formal model to minimise the energy consumption considering latency constraints, including a fog approach and nano-servers at the access point nodes (i.e., Multi-access Edge Computing).

In [176], a variability model for Dynamic Software Product Lines in Wireless Sensor and Actuator Networks is presented. In the model, a set of variation points are defined, which can be modified by human intervention at runtime. For example, in the proposed solution, a user could change the limits for a temperature alarm at runtime.

In [233], a component-based system, namely REMORA, is presented with the aim of providing system reconfiguration in sensor networks. The components in REMORA are based on event-driven communication. The operations provided by the components are described in XML, as are the events generated in the component. The adaptation of the components is done through parameters and, while the components and the framework in general are written in C, the part responsible for the dynamic deployment is developed in Java. The components are hosted in a repository and the system is configured according to the described model. However,

the proposal does not consider node mobility, replication/distribution of its components or network partitions.

In [15] a system for providing a WSN as a service, i.e., Network as a Service, is presented. For this purpose, the infrastructure is divided into three distinct parts: sensor network, gateway node and back-end, which is located outside the WSN. In this back-end or external server, an orchestration engine is deployed, which is responsible for deploying and configuring the services running in the WSN, communicating with them through the Gateway. The engine is model-driven. In this way, the services to be executed and the logic of the system are described in a XML model, which is interpreted by the engine to offer users the required services using the WSN as an IaaS. The system is implemented in Java web services and java scripts. However, the sensor network is static, both in the mobility of the nodes and in the specific nodes of which it is composed.

In [158], ODCWSN (on-demand customisable WSN) is introduced. The aim of this proposal is to allow defining the behaviour of WSN nodes by means of roles. The role is defined as a C program that specifies which sensors the node should use and how it should handle the data collected. The roles are generated by describing a scenario and downloaded to the corresponding nodes via their wireless connection at runtime. Specific data collection and aggregation operations are implemented as libraries that can be invoked by the roles.

The authors in [78] presents an architecture for customising the behaviour of intelligent devices at runtime. The architecture is composed of four layers. In the first one, FrontEnd, the user is provided with the necessary tools to interact with the architecture; in the Code Composition Layer, there are the components that define the behaviour of the device based on the composition of services and components; in the Artifact Contextualisation Layer, it is worth highlighting the use of plugins to standardise access to heterogeneous devices and sensors; finally, in the Smart Device Layer, the Runtime Environment component stands out, which is responsible for deploying the new applications created and launching them on the device at runtime.

3.4 Data Gathering with Mobile Elements

In modern monitoring systems, data collection and data gathering are inherent. However, it is still important to incorporate mechanisms or devices (e.g. mobile elements or nodes) that support the achievement of an adequate collection and/or gathering rate, or even decide where to send the data for processing.

Proposal in [165] tries to leverage the benefits that the cloud can bring by using mobile devices as thin clients to access remotely run applications, for data storage and processing. Certain eHealth applications require a computational effort for processing or data analysis. The authors propose a framework to identify whether to send or not data to the cloud in order to recognise what kind of activity (i.e., standing, walking, or running) the user is doing, and if so runs a composition of loosely coupled services in the cloud. The mobile device collects data from the accelerometer and hots a component (mobileCloudAdapter) to make the decision about configuration deployment. The cloud host a ServiceAdaptationModule to manage the runtime deployment of components and act as entry point of the mobile device. The evaluation measures performance by taking the CPU activity of the mobile device and of the server that act as cloud, and accuracy trade-offs.

Crowd-sensing is an IoT approach related with opportunistic MEs as gateway and whose movement cannot be controlled. In this context, research in [271] presents a cooperative framework to create sensing maps using human-carried or vehicle-mounted sensors. The area to be monitored is divided into different grid cells and a period is established in which they must be monitored. Each mobile device manages and maintains a table with the cells it has monitored and the time in which is performed. To reduce network over-head, a data fusion approach is applied. When two devices exchange information corresponding to the same grid cell and the same period of time, instead of maintaining and propagating both measures separately, these are merged in a single measure applying a fusion operator (average, maximum/minimum, sum or voting, inter alia) which avoids the propagation of redundant information. This cooperative monitoring framework has been used to construct noise, temperature or CO_2 level maps in a given urban area.

In [130, 131], the authors propose a rendezvous point approach for data gathering. The main objective is to define a set of sensor nodes that act as cluster heads, in this way the other sensor nodes send the data to them and the mobile elements only visit these nodes for data gathering. This will reduce the number of hops in the communication between the sensor nodes and the mobile elements. To define the cluster heads and the path of the mobile nodes, the problem is reduced to the Team Orienteering Problem. The cluster protocol is partitioned in four phases (control tree creation, clustering, ME trajectory estimation, and planning of routing paths). The evaluation is performed by simulations using Castalia and network lifetime is measured. No data aggregation operations are provided but the data sent are raw data.

In [76] a framework to improve data collection in a sensor network using mobile sinks (e.g., vehicles) is presented. The movement of these sinks is controlled and their only purpose is to collect data from the different sensor nodes. The sinks move along predefined paths,

although they can dynamically adjust their speed and pause to improve data collection in areas where events occur more frequently. Additionally, the sensor nodes adjust their transmission power to the path of the sink, in order to increase the lifetime of the WSN. The mobile sinks form a software network, managed by a centralized controller, which has a global view of the network and allows upgrading of the services running in the WSN according to the application requirements.

Research in [253] presents a data gathering scheme based on data fusion through a neural network called IDGS-DF. The field is partitioned in virtual grids where a cluster heads is selected to which data is transmitted by the cluster members. The election of cluster heads is based on the score of the nodes (the score is calculated based on residual energy and energy consumption, and it is proportional to the transmission distance) and these cluster heads include a pretrained neural network for data fusion to filter redundant data and to achieve energy efficiency. Mobile agents are used to send information through the predefined path. An evaluation has been carried out with a focus on energy conservation and increasing the lifetime of the network. It also evaluated the cost of the training process which decreases the more iterations occur.

In [48] a toolkit, entitled InCense, is presented which facilitates the creation and deployment of monitoring applications on Android mobile devices within eHealth scenarios. InCense supports mainly the monitoring of users automatically, through the sensors on the device, and/or through the participation of the user through surveys. InCense delivers: Dynamic reconfiguration which allows researchers to change the behaviour of the applications at run time; User-defined filters which enable the creation of custom components for data processing and monitoring; Data condensation permits the management representation and grouping of data; Automatically triggered and Event-triggered actions launch monitoring sessions when particular events occur.

The authors in [153] present a Reference Architecture for mobile crowd-sensing monitoring systems, focused on eHealth. This architecture is designed for those systems that demand the monitoring of environmental variables as well as the movement/position of the user. The architecture is composed of three main layers: (1) sensors, (2) data, and (3) microservices. The sensors layer is responsible for managing the sensors, activating and deactivating their corresponding listeners. In addition, it supports dynamic management of sensor sampling frequency, allowing them to be configured adaptively, fixed, timed or on-demand. Moreover, it supports data acquisition from the sensors either automatically, and voluntarily (i.e., information provided by the users themselves). The data layer is responsible for processing the data acquired by the sensors, carrying out operations of aggregation, filtering, compression and even a first

analysis. Finally, the microservices layer provides an application, which can be configured, to enable the reception of notifications and activity personalised feedback.

The authors, in [24], introduce a context-aware data fusion approach for health IoT applications as a generic vision for context-aware health systems.

Research in [129] proposes an application prioritisation system in order to reduce the delay, thus improving the data gathering waiting time of the highest priority applications at the expense of the waiting time of lower priority applications.

In [62], a data gathering system using drones (quadcopters) as mobile elements is proposed. The approach introduces an algorithm that calculates the route to be followed by the drone, taking into account the maximum distance it can travel according to its battery and considering the time required for the complete transmission of data to each node sensor.

In [53], the sensor data gathering problem is approached as the Team Orienteering Problem (TOP), in which a set of agents must visit a set of clients, and each client must be visited at least once by one of the agents. Note that, the translation of this problem to that of data gathering in sensor networks using mobile elements is straightforward. This is an NP-Hard optimisation problem, for which an optimal solution cannot be found in polynomial time. Therefore, the authors of this paper present a Lagrangian approach to solve the problem sub-optimally in reasonable time. For networks of 100 nodes and 5 mobile elements, they determine a route in approximately 115 seconds on average.

In [274], TCBDGA (Tree-Cluster Based Data Gathering Algorithm) is presented. In this proposal, a mobile sink periodically, starting from a station base, collects data from the source nodes. The mobile sink knows the location of all the source nodes deployed in the scenario. To carry out the data collection, the source nodes form a tree, based on their residual energy. Subsequently, the mobile sink will visit the root nodes.

The proposal in [200] presents a data gathering approach for in large-scale sensor networks using mobile elements. The nodes are organised into clusters and the mobile element collects data periodically from the cluster heads of each cluster. The location of each cluster head is transmitted to the base station and the shortest path through the locations is determined by a swarm optimisation algorithm.

In [8] is presented a clustering algorithm for data gathering by means of mobile elements, also called ferries by the authors. The proposal stands out because the cluster heads are defined on the basis of the tour defined by the mobile element. In this way, the existing mobile elements of the scenario can be used to gather data whilst slightly influencing their movement pattern, since the mobile element must make a stop at each of the cluster heads defined according to the proposed system.

3.5 Discussion

Although the proposals reviewed approach a common scenario, i.e., monitoring systems, they address different yet related issues. To the best of our knowledge, there is no proposal that offers a comprehensive solution for the set of issues that this thesis addresses.

With respect to frameworks for the design and development of monitoring systems, although the benefits of visual paradigms to facilitate and bring system design and development closer to domain programmers, or even to the end user, are well-known [59], there are few proposals that offer visual development systems in sensor network or IoT scenarios. Those that address the problem usually consider that the sensors are connected to Internet, or only allow the behaviour of the sink node (gateway) to be modified. However, this excludes configurations based on mobile elements, ad-hoc connections, or dynamic network topologies, reducing the flexibility of the proposals.

In parallel, there are several proposals that address monitoring system configuration upgrade at runtime in WSN nodes. This issue is of relevant importance due to monitoring systems complexity and their dynamic environments. However, no proposals have been found that use mobile elements, not only for data gathering, but also to propagate from minor system modifications to major upgrades to those nodes. This is of particular relevance in large area sensor networks, thus avoiding the need to deploy a fixed communication infrastructure and therefore reducing the cost of the solution.

Mobile elements are a common resource in sensor networks for data gathering. Two types can be distinguished, (1) those that are naturally part of the scenario (e.g. a hiker in a natural park, or a migratory animal), and (2) those that are introduced as an additional element for the sole purpose of data gathering (e.g. a drone). The main difference between these two types is that, in the former, the trajectory can not be modified or controlled. This affects the data gathering, making it an opportunistic data gathering in which some delay may occur from the time the information is generated to the time it is received by the sink node. However, as these elements are already naturally present in the scenario, they do not represent an additional cost for the system. In the second type, when the trajectory of the mobile element can be established, the main challenge is to optimise data gathering while reducing the distance travelled. In this context, remote vehicles such as drones (terrestrial or aerial) are often used. While they may offer better results for data gathering, in terms of amount of data gathered and delay, the main disadvantage of this approach is found in the physical limitations that these elements encounter in moving over complex terrains, in the case of terrestrial drones, and in the weather conditions (wind or rain) in the case of aerial drones.

Finally, and from a general point of view, it should be noted that most of these proposals provide an ad-hoc solution for a specific monitoring scenario, making it difficult to transfer or adapt it to other monitoring scenarios other than those for which they were initially designed. Furthermore, the inclusion of preprocessing operations in the network should be highlighted, as only a few of the proposals reviewed include this functionality. Nonetheless, this can be considered as a key option to reduce the consumption of network resources, improve data gathering by reducing transmission time, and, ultimately, to fulfil the property of localised scalability in the system.

The solutions proposed to address these problems must be flexible, modular, versatile, elastic, extensible, customisable and configurable. To this end, the proposal must have the ability to adapt and self-adapt to specific situations in dynamic environments accordingly.

3.6 Summary

This chapter has reviewed the existing proposals in relation to the main objective and the goals thesis. These are divided into three different areas: (1) design and development of monitoring systems in order to increase reuse for different application domains; (2) facilitating autonomous deployment in the WSNs minimising human intervention, and supporting reconfiguration at runtime; and (3) data gathering in WSNs with mobile elements.

A total of 30 proposals published between 2012 and 2021 have been reviewed. Of these, 19 belong to scientific journals, while the remaining (11) belong to communications at international conferences.

Although the proposals reviewed together address, in general, the different problems addressed by this thesis, it has not been found a proposal that offers a comprehensive solution.

Features such as run-time adaptation and self-adaptation; abstraction in the development of the system through visual paradigms; in-network preprocessing to achieve localised scalability; and taking advantage of the mobile elements that can be found in the scenario, both for data gathering and for the propagation of modifications and upgrades in the system, are desirable features that a solution that aims to fulfil the objectives proposed in this thesis should integrate.

Part II

Monitoring Systems Developed and Studies Conducted

Chapter 4

Monitoring Systems

Chapter Abstract

This chapter describes detailed information about the application domains under study and for which we initially contributed to the development of monitoring software systems and selection of hardware devices.

During this initial stage of the thesis development, numerous meetings and collaborations with experts (psychologists, doctors, nurses, midwives, educators and therapists) were held. These experts helped us to identify the requirements for providing adequate functionality considering both patient monitoring and care in general.

The development of monitoring systems includes data collection from non-intrusive sensors and wearables, the combination of subjective and objective information, data preprocessing actions and post-processing analysis according to the needs. It has been identified that all these systems share several similarities.

Chapter Contents

4.1 Introduction

Initially, a monitoring system for the Sleep Apnea Hypopnea Syndrome (SAHS) was developed. Its design already followed guidelines and decisions were taken into account that distinguished it from traditional ad hoc monitoring systems. This system served as a preliminary basis for further developments related to other cases.

Starting from the basis of Sleep Apnea Hypopnea Syndrome (SAHS) monitoring system, we then addressed the Systemic Lupus Erythematosus (SLE) case, and subsequently we continued with Pregnant Women with Small for Gestational Age (SGA) Foetuses case. Therefore, three have been the mainly study cases addressed in greater depth during this stage. At the same time, it has helped us to realise that particular case studies even application domains share numerous similarities mainly related to functionality. In particular, and also in relation to the functionality of the system, the collaboration with the domain experts was key to determining the requirements and viability of the systems.

In addition, monitoring of the environment has been included following the study of certain proposals that began to take into account environmental factors (e.g., temperature). There were no preliminary studies in these specific application domains, however, the expert contributors agreed that certain factors could be relevant, even conditioning factors for the health and emotional state of patients. Furthermore, it should be noted that this is one of the ways to ensure the ecological validity of data collection.

The architectural design of the monitoring systems developed supports the specialisation of services. This approach makes the systems more flexible because adding new devices is easy and immediate, as there is no need to develop new software modules but to specialise the existing services. With regard to the conduct of studies both objective (i.e., physiological and context information) and subjective data are collected with non-intrusive sensors or wearables, and standardised questionnaires, respectively.

The chapter is organised as follows. Relevant aspects of the diseases on which the thesis focuses at this initial development stage, as well as the most affected patients, motivation, and context-specific information, are available in Section 4.2. Details of each of the systems proposed and the studies conducted can be found in Section 4.3. Finally, some relevant considerations about functionality, common aspects identified from the development, and technologies are explained in Section 4.4.

4.2 Application Domains

Several application domains have been considered during this thesis. This section briefly describes their characteristics, symptoms, risks and consequences on patients' lives, variables that can be monitored and common practices. The help of experts in each of these specific domains (doctors, nurses, psychologists, midwives, educators and therapists) has been a key element in the development of these systems.

Environment monitoring can be considered as a complementary domain of application in multiple situations, including those mentioned above.

4.2.1 Sleep Apnea Hypopnea Syndrome (SAHS)

The Sleep Apnea-Hypopnea Syndrome (SAHS) [92] is classified in the dyssomnias group and is characterized by drowsiness, cardiorespiratory and neuropsychiatric disorders, that lead to repeated episodes of obstruction in the upper airway during sleep. It implies high blood pressure, a serious decrease in quality of life, traffic and workplace accidents or even die asleep. Nowadays, between the 2-5 % of the world's population suffer from this syndrome and from which close to 90-95 % have not been diagnosed [268]. This disorder affects people of all ages, children and adults, but the symptoms and treatments are different for both. In general, the probability of developing this disease in adulthood is higher in the case of men. When men reach the age of 40 and woman reaches menopause, tends to equalize the probability. Other factors that increase the probability of developing this disease are overweight, hypertension, abnormalities or defects that can affect the upper airway, among others. Some of the most common symptoms of SAHS are asphyctic episodes, observed apnea, abnormal movements and frequent awakenings.

Usually, the diagnosis of SAHS requires to perform a test in a specific room called Sleep Room (nocturnal polysomnography). One of the great disadvantages is that such installations are scarce for the high demand that exists, therefore, it leads to a long waiting list of patients. The sleep test requires that the patient remains asleep for several hours. Likewise, the patient is in a strange environment so in many cases makes it more difficult for sleeping, which leads to repeat the test with the repercussions that this have on the own patient and on the waiting list. Furthermore, the sleep room has a sophisticated, static, heavy and expensive medical equipment which allows detailed studies of patients. In addition, a specialized medical staff is required to place the sensors in the body of the patient. This medical staff also monitor the patient while he or she sleeps, and in the case that there are any problem with the equipment or

the patient needs an urgent medical attention, because he/she is in a critical state, the medical staff can intervene. At present the nocturnal polysomnography is the most reliable study used to detect whether a person suffers from this disease [56].

4.2.2 Systemic Lupus Erythematosus (SLE)

Systemic Lupus Erythematosus (SLE) is a chronic autoimmune disease mainly affecting young women which is associated with several clinical symptoms such as pain, fatigue and emotional distress that impair the quality of life [167, 251]. However, SLE does not present specific factors to consider for monitoring and diagnosis its unknown and uncertain etiology. Therefore, the appropriate medical tests for its study and the configuration of the devices require changes in short periods of time. Nowadays, the affected population in the world is between 0.02-0.07% and there is a prevalence in woman compared to men on a scale 9-1 [189]. Also, it is important to emphasize that the SLE is not an isolated disease, commonly there is a high prevalence of Fibromyalgia (FM) [157] and sleep disorders [54].

The very few studies that have examined sleep in this clinical population have observed sleep alterations related to insomnia [180, 112]. The two unique studies carried out by using objective measures (polysomnography referred to by its acronym PSG) revealed alterations in sleep architecture, microstructure and continuity in SLE patients (e.g., a decrease in total sleep time, delayed sleep onset, sleep fragmentation, and reduced delta sleep) [244]. To date there are no studies using actigraphy. Actigraphy is a modern technique that is less intrusive than PSG. This allows experts to assess the wake/sleep cycle from physical activity and patient movement.

4.2.3 Pregnant Women with Small for Gestational Age (SGA) Foetuses

Small for Gestational Age (SGA) is a term used when a foetus is small considering the number of weeks of pregnancy. The diagnosis requires to know the gestational age and measurements of height, weight and head circumference of the foetus. In developed countries these cases reach between 5-10% [91, 31]. These babies have a birth weight below the 10th percentile are considered SGA, in other words, most babies weigh 2.6 kg at birth, therefore those weighing less than 2.5 kg are SGA [106, 243]. It should be noted that these babies have higher morbidity and risk of suffering from certain diseases during childhood and even as adults and mortality during neonatal period is higher with respect to appropriate-for-gestational-age babies [91, 31].

Emotional management problems of the parents increase the risk of having a SGA or premature baby [104]. Moreover, stress during pregnancy causes alterations in the vascular,

neuroendocrine and immune systems of the foetus and increases the likelihood of foetal growth retardation and preterm birth [166, 229]. In addition, when mothers experience stress during pregnancy, there is also a prevalence of neurobehavioural underachievement in children from an early age to adolescence, and there is also evidence of emotional and cognitive problems [6, 32].

4.2.4 Environmental Monitoring

An adequate management of context information is a challenge due to its complexity and most proposals are limited to establishing empirical relationships between a few certain aspects. However, certain environmental factors such as luminosity, noise, temperature, inter alia can affect or negatively influence the state of health of people. In addition, environmental factors are framed as contextual information, and context in turn encompasses everything around us (i.e., home, work, recreation area, or activity performed at a given moment, he/she is watching the TV while also filling a form, etc.).

Therefore, the relation between physiological variables and environmental factors at a given moment can help to determine causes, alterations, the occurrence of specifics symptoms (frequency of arousal, time intervals, sleep quality, corporal indisposition, insomnia, etc.). Also, this information allows experts to determinate if there are guarantees on the *ecological validity* of the research [209, 63]. Schmuckler [209] points out that the concept of *ecological validity "has typically been taken to refer to whether or not one can generalize from observed behavior in the laboratory to natural behavior in the world."*. Previously, Bronfenbrenner [43] defines *ecological validity* concept as the measure representing whether the environment perceived by the users is the environment to be perceived and assumed by the experimenter. This concept is highly related with multimodal integration, where there may be disruptive effects between data sources or conflict situations due to the receipt of inconsistent data, both in the laboratory or natural environment [63].

4.3 Systems Developed

Starting from the basis of system for the Sleep Apnea Hypopnea Syndrome (SAHS), the first system developed, a posteriori, two more systems have been developed and an additional one designed. In this section, it can be found the main functional requirements of each of them which helps us to define what the system should accomplish [227].

It also includes a summary of the most relevant details related with its design, architecture, data base and management mechanisms as well as specific aspects of the sensors that are part of the network infrastructure, and technologies used considering their suitability to the case.

In addition, some of the results and conclusions on the evaluations carried out are described.

4.3.1 Service-Oriented Monitoring Systems Assisting Diagnosis and Treatment of the Patients with SAHS Symptomatology (SMODIAT)

A system for remote monitoring and diagnosis of patients who could suffer the sleep apnea-hypopnea syndrome trying to emulate the sleep room of a medical center with low-cost sensors was developed [20]. One of the main objectives of the system was to reach a major number of people at the same time and reducing the long waiting-lists for the test of polysomnography.

The medical team could perform the monitoring at the patient's home even the patient himself could place the sensors. The test can be repeated as many times as necessary due to the availability and low cost of the equipment, and the patients would be in a familiar environment which favours their sleep to be closer to what they regularly have.

The functional requirements specification of the software system are summarised as follow:

- **FR.1** Three types of users (medical specialist, family member, patient) should operate in the system.
- **FR.2** The system should support the addition of sensors and/or ergonomic devices to monitor patients' physiological measurements (e.g. heart rate, blood oxygen saturation, patient movements, body temperature, sweating, electrical activity of the heart, inspiratory flow, etc.). Note that these devices possess heterogeneous characteristics.
- **FR.3** Sensors should be individually assigned to each patient.
- **FR.4** The medical specialist should be able to monitor the patient in real time.
- **FR.5** Patient information (name, date of birth, gender, weight and height) must be stored and must be accessible at any time. The patients should be able to be registered within the system by the medical team which should be able to modify its information.
- **FR.6** Patient information gathered with the sensors should be accessible to specialist at any time. It should be shown graphically in order to observe variations over time. Moreover, maximum and minimum values reached, arithmetic average and standard deviation (i.e., data processing) should be registered and should be available for consultation. The medical team should be able to make notes at sessions.

- **FR.7** The information gathered should be stored separately by sessions and each session should be available for consultation (e.g., report format) at any time. Sessions should include start and end date and time.
- **FR.8** It should be possible to consult the information about all the patients.
- **FR.9** Specialists should be able to configure alarms by setting thresholds (maximum and minimum). When a thresholds is reached, a notification must be sent to the medical specialist.
- **FR.10** The specialist should be able to configure how he/she receives notifications (i.e., type of sound, sound volume, with message or both, and with the option to activate or deactivate them). The notifications should be sent regularly as long as the situation remains unchanged and conditions no longer trigger it.
- **FR.11** The family members should receive a simplified notification of the patient's condition.
- **FR.12** The functionality should be available via mobile device.

Design

The system proposed and developed follows a SOA approach, structuring the functionalities of the system in a hierarchy of layers. In the main layer, the service platform is framed, in which the following services are mainly included: 1) connection and communication management services between each sensor and/or device with the corresponding application; 2) data monitoring management services; 3) data and information storage services; 4) data and information processing services; and 5) data/information processing and analysis services.

The system offers specific functionality through a specific and adapted user interface for each of the stakeholders involved. The architecture (Figure 4.1) of the proposed system is divided into three subsystems:

- *Patient monitoring subsystem* monitors the physiological measurements of the patient using a WBANs, in real time and it sends the gathered data to a central unit, called monitoring application. The monitoring application has the computational capacity to receive, store and process the data gathered locally and/or externally.
- *Specialist supervisor subsystem* provides information of a patient to the specialist or specialists team, preprocessing, processing information, reports of sessions and notes. It

allows to set up separate alerts for each patient or group. It also notifies if a sensor has been disconnected.

- *Family monitoring subsystem* provides information to a patient's family member through their own mobile device. It also notifies if a sensor has been disconnected. This user can customise how he/she want to receive notifications.

Two main services are to be highlighted, namely *Database Management* and *Patient Information Management* services which follows the standards of web services such as SOAP and WSDL. These standards were used to provide an uniform access and independence from platform. In this way, different devices with different properties, at hardware and software level, can access the *Database Management* service and this service supports interoperability with other systems, services, and applications. The *Patient Information Management* requests data to *Database Management* service to carry out data processing with the objective to provide more complex information (i.e., processing data) that may require an intensive computing. These services can be run on a local server or in the cloud.

If a disconnection occurs during the monitoring session, and it has no connection available with the service, it stores the information collected locally, to subsequently send it, when the connection is reestablished, applying synchronization.

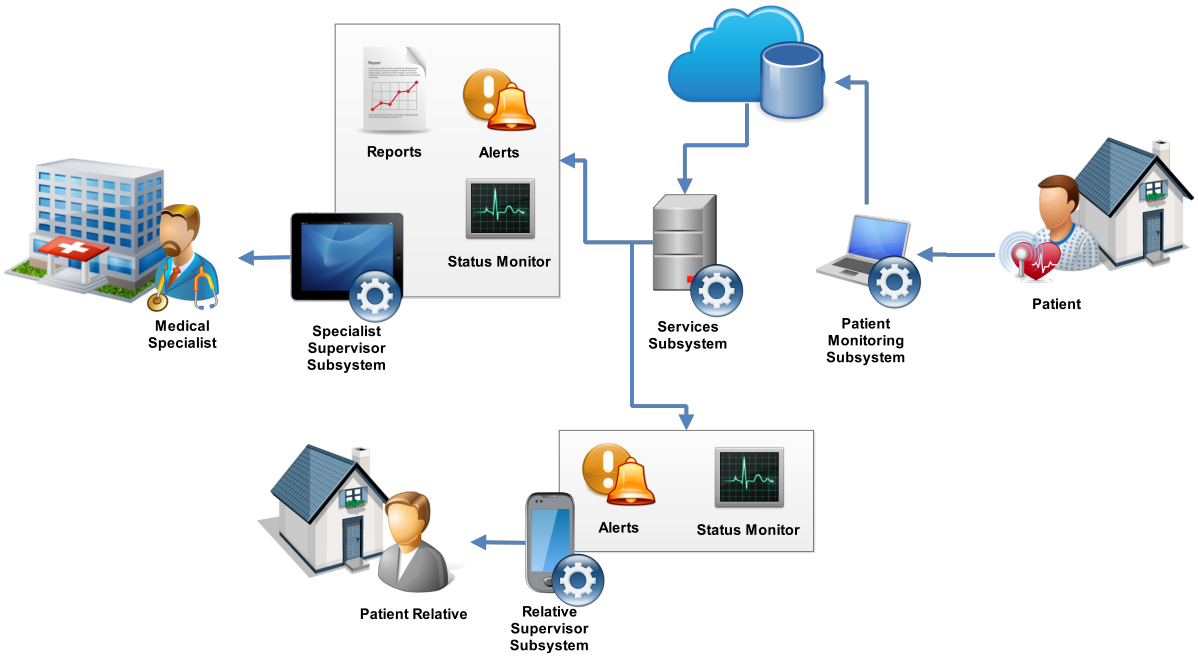


Fig. 4.1 *SMODIAT architecture.*

The design and development of the proposal follows a generic and distributed approach. The reason is that the core functionality is framed within the general platform of services with the aim of acquiring the principles of autonomy, composition, abstraction, low coupling, platform independence and reusability, and facilitating scalability. Each service can be specialised to provide an adaptable system, enabling the creation of dynamic applications to solve a multitude of problems of varying complexity.

Three simple problems, that could be directly solved, are shown below as example:

- **Heart rate** component can detect an abnormal heart rate. The range varies depending on the age of the person, his/her gender, whether he/she is an athlete or a sedentary person. A regular range is $L_1 < 45 - 100 < L_2$ Beast Per Minute (BPM) where L_1 or L_2 are measurement outside the range limits, either lower or upper.

- **Blood oxygen saturation** component can detect low oxygen saturation values which are common in patients suffering from apnea or lung problems. An abnormal blob oxygen saturation is $L < 92\%$ where L is a value bellow the recommended value.

- **Awakenings** component can detect an unusual number of awakenings.

Technologies such us Arduino and NexGen Ergonomics sensors were used. The applications were developed with Android Studio and tested on tablets and smartphones with Android Operating System.

4.3.2 Mobile System for Monitoring the Environment (CEnMO App)

A mobile Health (mHealth) system for objective monitoring of the environmental context has been developed. Environmental factors (luminosity, environmental temperature, noise, etc.) could influence or alter the physiological variables (hear rate, temperature, blood pressure, sweating, etc.), and therefore the patient health state (exhaustion, depression, sleep disturbance, etc.) [266, 81].

The functional requirements specification of the software system are summarised as follow:

- **FR.1** Monitoring environmental luminosity, temperature, humidity, pressure and noise adding a timestamp to identify when each measure is taken.

- **FR.2** It should be possible to configure which measurement to collect.

- **FR.3** Sample rate should be configurable (as a software functionality).

- **FR.4** Data should be able to be stored locally or sent to an external server.
- **FR.5** Data should be preprocessed. Cleaning of invalid data from the synchronisation period and aggregation operations (maximum, minimum, and arithmetic average) on a specific dataset must be provided.

Design

The developed mHealth system follows a services and microservices approach, and it includes a mobile application namely Context Environmental Monitoring (CEnMO). Its architecture is shown in Figure 4.2 and a sample of CEnMO App can be seen in Figure 4.3. CEnMO must be installed on mobile devices with built-in luminosity (Lx), temperature (°C), humidity (%) and pressure (mbar or hPa) sensors, and microphone. Afterwards, simply place the user can place the mobile device in a specific location (a room, bedroom, hallway, living room, etc.) and start the application to begin collecting data.

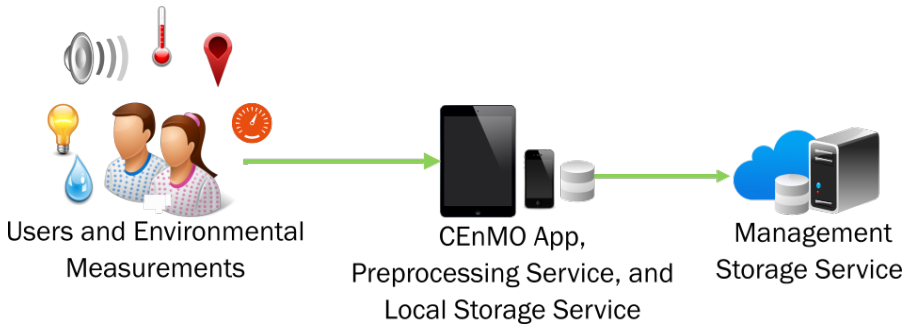


Fig. 4.2 *mHealth architecture.*

The system covers the needs of the requirements, specifically it addresses the management of the configuration of the sensors integrated in the device (activating or deactivating them in a specific device and setting a sampling frequency). Input data is also preprocessed, and in particular, the microphone data is transformed into a quantifiable unit of measurement or noise level in decibels (dB). The system works with noise level, not with sound recordings in order to preserve the privacy of the users.

The proposal of the mHealth is flexible, configuration parameters can be reset or modified during monitoring and allows adaptation for the management of different types of information. This provides a valid system not only for monitoring the environment, but also supports the monitoring of other types of context.

CEnMO App was developed with Android Studio and tested on tablets and smartphones with Android Operating System.

(a) *List of users.* **(b)** *Start monitoring.* **(c)** *CEnMo users, monitoring, data management and dashboard.*

Fig. 4.3 *CEnMo app.*

Study Conducted

Based on CEnMO (Section 4.3.2) and SMODIAT (Section 4.3.1), the design of a generic model which includes monitoring of physiological variables of patients and environmental factors is proposed. These systems consider both kinds of measurements in order to assess objectively the state and evolution of the patients ensuring the ecological validity of data. To accomplish this goal, the proposal makes use of services and microservices, whose design is based on components in order to be able to adapt a same solution to several diseases with some common disorders, specific equipment, and environmental situations of the patient.

Figure 4.4 shows a specific component diagram that has been used in the objective analysis of Sleep Quality on SLE patients using actigraphy and mHealth systems [22]. The motivation for the work was looked at sleep quality of SLE patients based on more objective information provided by actigraphy and mobile systems, check the ecological validity of the data and how environmental conditions and factors can affect sleep quality. In traditional methods, the information for assessing sleep quality is obtained through questionnaires but this study introduces a novel method that combines subjective (standardised questionnaires) and objective information using actigraphy wristband and environmental conditions through the easy to use and non-intrusive mHealth system. The method provides some mechanisms to detect how sleep

hygiene could be associated directly with the sleep quality of the subjects, in order to provide a custom intervention to SLE patients.

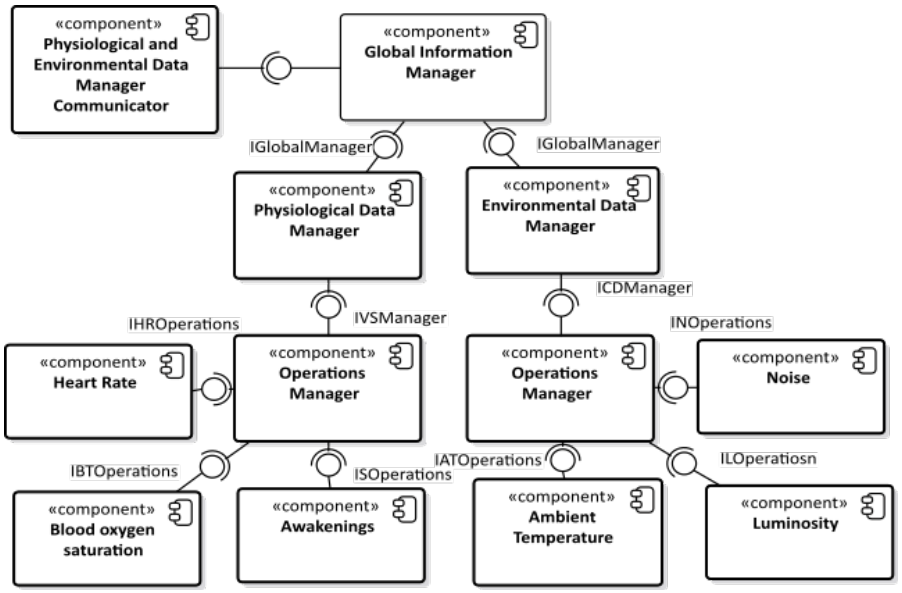


Fig. 4.4 *Component model of physiological and environmental measurements manager service for mHealth systems.*

The system was evaluated and a comparative analysis between SLE patients $(n = 9)$ and healthy controls $(n = 11)$ for psychological variables, actigraphy, and variables related to environmental conditions was performed. The results show that luminosity has a significant correlation with time in bed $(rs = 0.80, p < 0.01)$ and temperature correlated significantly with actual waking time $(rs = 0.85, p < 0.01)$, sleep efficiency $(rs = -0.81, p < 0.01)$ and sleep latency $(rs = 0.92, p < 0.01)$.

4.3.3 mHealth system to assist pregnant women through a psycho-educational programme (mPOP)

A mobile Health (mHealth) system, namely mPOP, to provide a psycho-educational programme in order to assist pregnant women with SGA foetuses and their partner was proposed [23]. In the case of pregnancy, the mHealth systems facilitates self-report of emotional health (e.g., depression, anxiety, stress, wellbeing, motivating changes or supporting therapeutic interventions). There are also concerns related to the care of the baby, and even more worrisome ones when they are small for gestational age or if there are malformations.

The functional requirements specification of the software system are summarised as follow:

- **FR.1** To provide a tool for designing multiple contents integrated psycho-educational programme to assist pregnant women and their partner.
- **FR.2** Different types of content should be able to be added and organised.
- **FR.3** It should be possible to create different programmes with different contents, modify them or reorganise, delete them and each programme must be able to be used by different users.
- **FR.4** It is necessary to be able to adapt/modify at any time the contents or the whole programme associated with a specific user because their needs may change due to their state of health and intervention may be required. These actions would be carried out by medical specialists such as midwives, doctors and psychologists.
- **FR.5** There should be no restrictions to follow the programme, all content should be open. The user will be able to choose which tasks to perform and/or content to display.
- **FR.6** Each task could have one or several contents.
- **FR.7** The content should support different formats of resources such texts, videos, music and links to internal or external resources.
- **FR.8** It is necessary to store information of system users.
  - Mother: identifier, name, age, gender, cohabitation, relationship time, education level, job, gestation week, first contact data, number of previous pregnancies, number of previous abortions, number of children alive, natural pregnancy, obstetrics risk, induced labour at on set, characteristic of labour.
  - Partner: identifier, name, age, gender, cohabitation, relationship time, education level, job.
  - Psychologist/Doctor: identifier, username, specialty.
- **FR.9** Physiological data that help to detect stress or other types of alteration (e.g., nervousness, fatigue, inter alia) should be collected automatically in a non-intrusive way. The physiological data to be considered are acceleration, blood pressure, heart rate, heart rate variability, electrodermal activity, and body temperature. An event marker and a time marker should be included to record specific points in time.

- **FR.10** Pregnant women and their partners should have access to the programme via a mobile device.
- **FR.11** The psycho-educational programme must exist in order to be associated with a user.
- **FR.12** Data collected from questionnaires should be send to an external server.
- **FR.13** Data should be preprocessed. Cleaning of invalid data from the synchronisation period and aggregation operations (maximum, minimum, and arithmetic average) on a specific data set must be provided.

Design

The requirements of the system were defined in collaboration with medical specialists such as psychologists, doctors and midwives. The mPOP architecture can be seen in Figure 4.5 and its main aspects are detailed below.

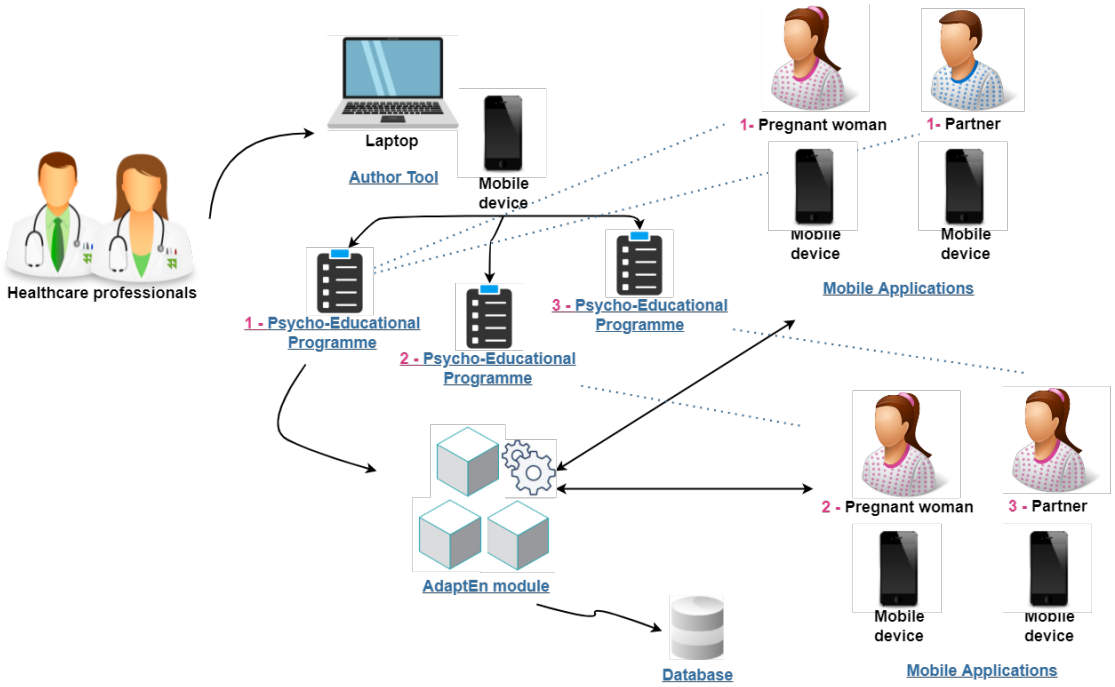


Fig. 4.5 *mPOP architecture.*

- A mobile application has been developed with a very flexible structure that organises tasks in turn include optional explanations or recommendations, activities, questionnaires

and links to supplement the recommended basic information. These tasks are designed to be carried out by pregnant women or their partners. The contents are grouped thematically and planned to be carried out as simple day-to-day tasks.

mPOP's design allows it to housed different intervention programmes. As example, in Figure 4.5, three different programmes (i.e., 1 - Psycho-Educational Programme, 2 - Psycho-Educational Programme, and 3 - Psycho-Educational Programme) have been included. In this way, different aspects can be specifically addressed, and others can be emphasised according to the needs of the users while medical supervision specialists is continuous.

- New psycho-educational programmes can be added. It is also possible modify them in runtime based on the user's needs and health state. The contents are designed and organised by the specialists, but the uploading or modification is automated as it is managed by the AdaptEn module (Subsection 4.3.3). AdaptEn module in turn connects to an external database (called MoSysDB) which stores the information.

- A generic interface was designed to accommodate different programmes irrespective of their content, type and number (Figure 4.7). The views are dynamic and only the part that has been modified is reload.

- An author tool has been developed to manage users' pregnancy information and to manage the association with intervention programmes.

- Personalised psycho-educational programmes will be available on users' devices immediately after loading or modification. Therefore, different couples, even different partners, may have different programmes.

- Mothers and their partners can access to the psycho-educational intervention programme through their mobile devices, and complete it at any time and from anywhere.

- During the development of the programme, questionnaires will be displayed with one or more questions related to specific tasks. The completion of tasks and questionnaires are monitored in order to analyse user progress and programme satisfaction. The programme can also be adapted considering the results of completion and the mother's health state.

The mPOP system proposed and developed follows a microservices based approach. The MoSysDB database scheme is generic with the aim of being able to host different intervention

programmes for several health domains, not only related to small-for-gestational-age pregnancies or babies. MoSysDB structure allows the development of monitoring systems and database-driven mobile applications making it possible to change the internal content of an already installed application, without having to reinstall or update it. MoSysDB also covers the representation of various types of content and formats required in this area in order to be able to anticipate the needs to store a type of content not known a priori in the early stages of the design and development of a specific system and/or application.

The mPOP system model (Figure 4.6) comprises mainly eight entities:

- *Baby, parent, mother and healthcare professionals* entities (blue background).
- *Applications and author tools* entities for parents to access to the psycho-educational programme and for healthcare professional to manage the programme (green background).
- *Adaptation Engine (AdaptEn)* entity represents the module that carries the load of the intervention programme the first time, and then making changes to adapt it once deployed (yellow background).

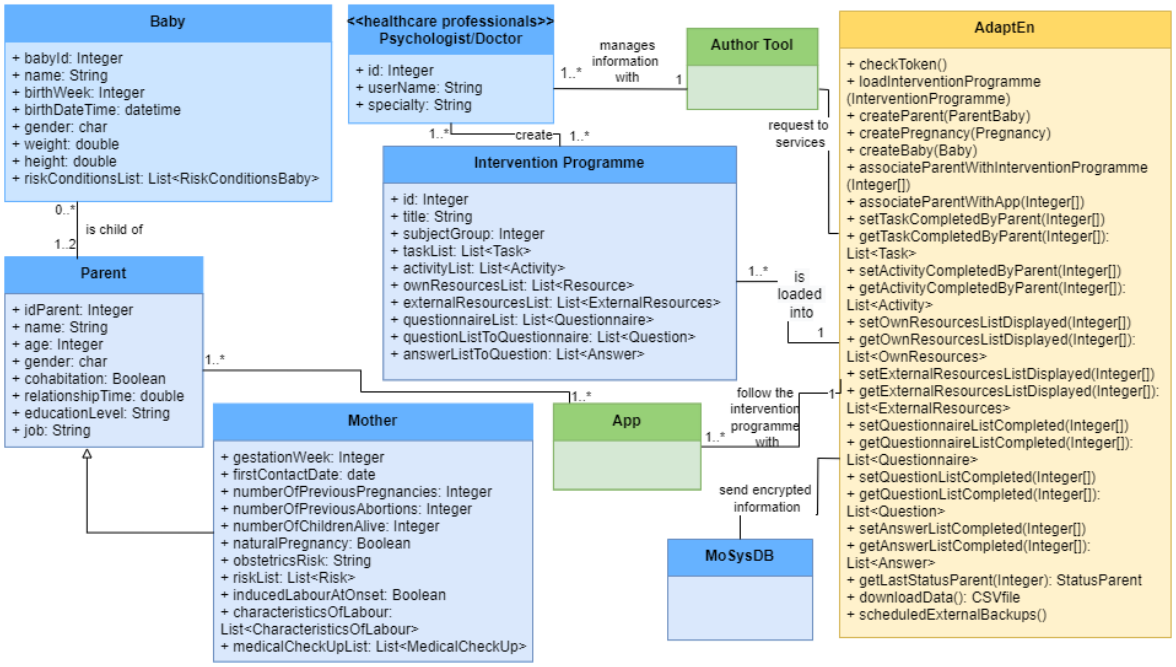


Fig. 4.6 *mPOP system model.*

Currently, the mPOP system is supporting three psycho-educational programmes as independent applications (i.e., VIVEmbarazo (Figure 4.7), VIVECrianza Hospitalización and

VIVECrianza No Hospitalización) were deployed. These applications are easy to install on the user's own device.

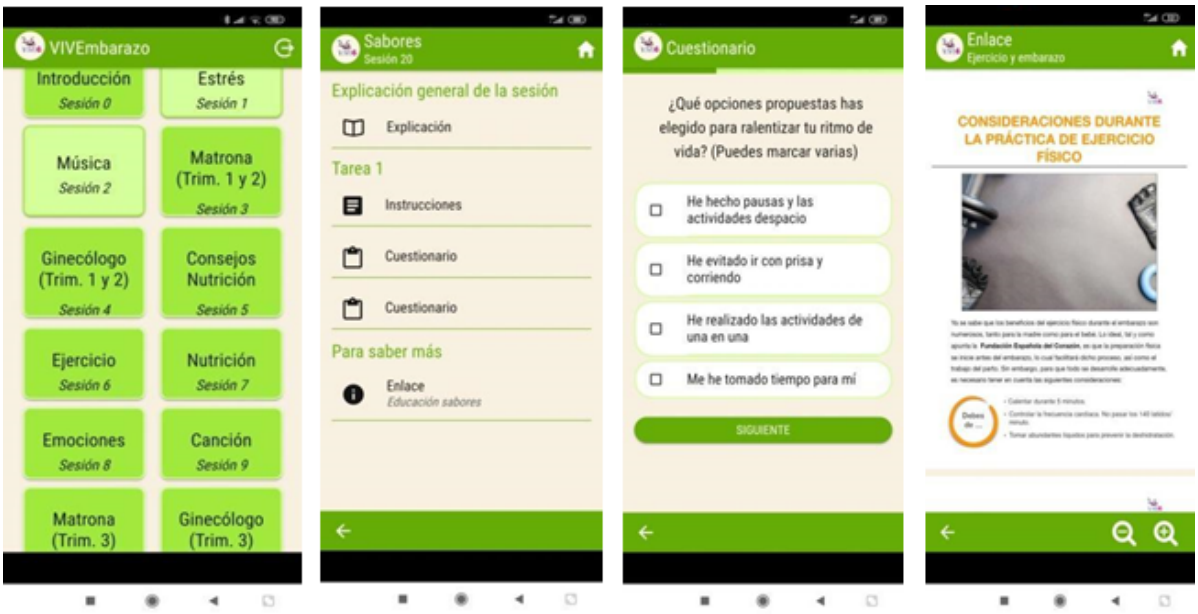


Fig. 4.7 *App VIVEmbarazo.*

Along with the mPop system, the E4 wristband (Empatica bracelet) [73] to monitor the physiological measurements required within the specification has been included. The E4 wristband collect electrodermal activity (EDA) expressed as microsiemens (μ S), blood volume pulse (BVP), heart rate (HR), inter beat interval (IBI), acceleration (range -2g, 2g), body temperature expressed on $^{\circ}$ C.

The applications and author tools were developed with Angular.js framework and Ionic framework. The applications were tested on tablets and smartphones with Android Operating System. The author tools were tested on smartphones and computers.

Study Conducted

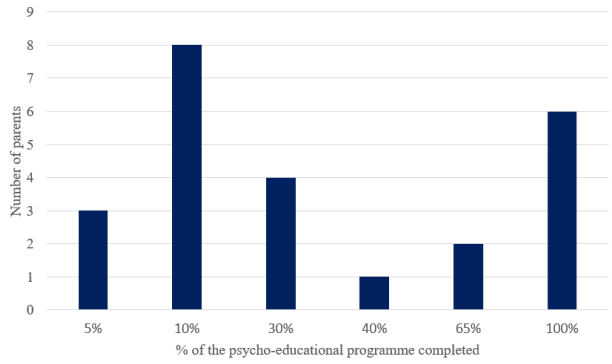
A proof of concept with the VIVEmbarazo App was carried out in order to help pregnant women and their partners through a psycho-educational programme. VIVEmbarazo includes more than 20 tasks organised in four pregnancy areas: medical advises, health care, communication with the foetus (stimulation), and emotional management.

Preliminary results from 24 users who have participated in the program provide data on the degree of follow-up of the program and the areas of greatest interest (Figure 4.8).

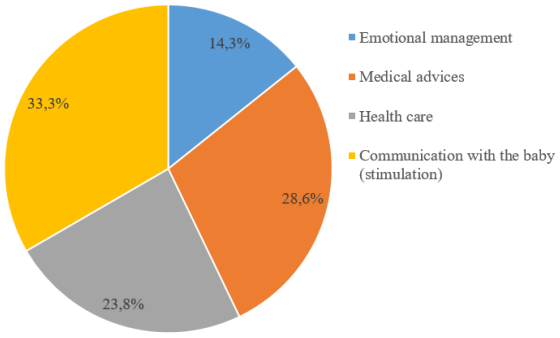
The left graph shows a parental engagement (Figure 4.8a) in the intervention programme. Of the total number of users, 63.5% have completed more than 65% of the programme. Some mothers abandoned the programme before completed it because the birth of their premature babies.

The right graph show the percentage of engagement (Figure 4.8b) in the tasks grouped by the four main areas. Communication with the baby (stimulation) area followed by medical advice area were the ones that the parents completed the most.

All the users provided positive feedback about VIVEmbarazo usefulness.



(a) *Parental engagement.*



(b) *Pregnancy areas.*

Fig. 4.8 *Preliminary results of mPOP system.*

Adaptive Engine to Manage Structural and Dynamic Content in Apps and Systems (AdaptEn)

AdaptEn module is hosted in a cross-cutting services layer. Clients connect to the server, make a request (HttpRequest) and the response message is encapsulated in an HTTP-response. Security aspects are covered by the checkToken service which establishes a secure connection and encrypts the information transmitted over the network.

The AdaptEn provides support for 1) creation of intervention programmes; 2) associating the most appropriate intervention programme to parents, taking into account their specific needs; 3) uploading a programme into an application; and 4) monitoring the completeness of each parent's tasks and download the answers of the completed questionnaires. We have made three activity diagrams to describe the behaviour of AdaptEn using control flow:

- The *creation of psycho-educational programmes* (Figure 4.9) requires that healthcare users uploading a programme (which already includes tasks, general resources and specific information, activities, questionnaires and possible response options, and other

additional resources). The AdaptEn module creates the programme and the app that host it. It is possible to create as many programmes as necessary, as well as to generate a mobile application for each of them with minimal human intervention. Moreover, the reassignment of new programmes without the parents having to reinstall the application is feasible.

- *Associating a programme to a parent* (Figure 4.10) requires that healthcare users to designate through the authoring tool which programme is assigned to which mother or couple. In this authoring tool, it is necessary to search for the parents in order to make the assignment. If parents do not yet exist, they must be created for the allocation of the intervention programme.

- *Loading an app on the associated parent's mobile device* (Figure 4.11). The application requires to AdaptEn the data to be loaded into each fragment right when the user accesses. Once the data is obtained, the view is created and displayed to the user. This makes it possible to change intervention programmes at runtime and the change is imperceptible to parents as it does not require any action on their part, but they would already directly visualise the adapted programme.

  Navigating the application requires that AdaptEn consults MoSysDB. In addition, when it is a questionnaire, and parents answer it, such answers are stored in MoSysDB.

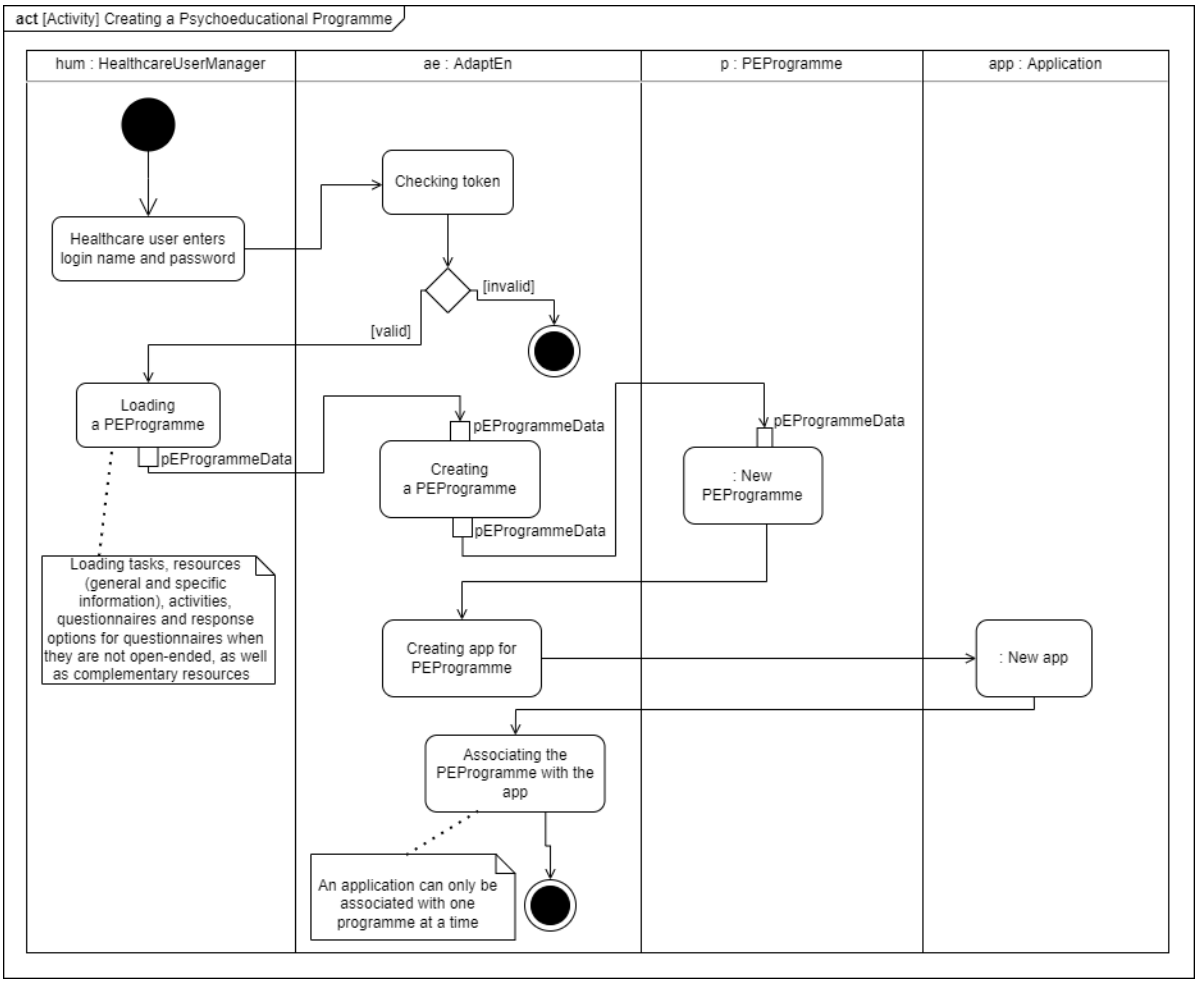


Fig. 4.9 *Activity diagram for creating a psycho-educational programme.*

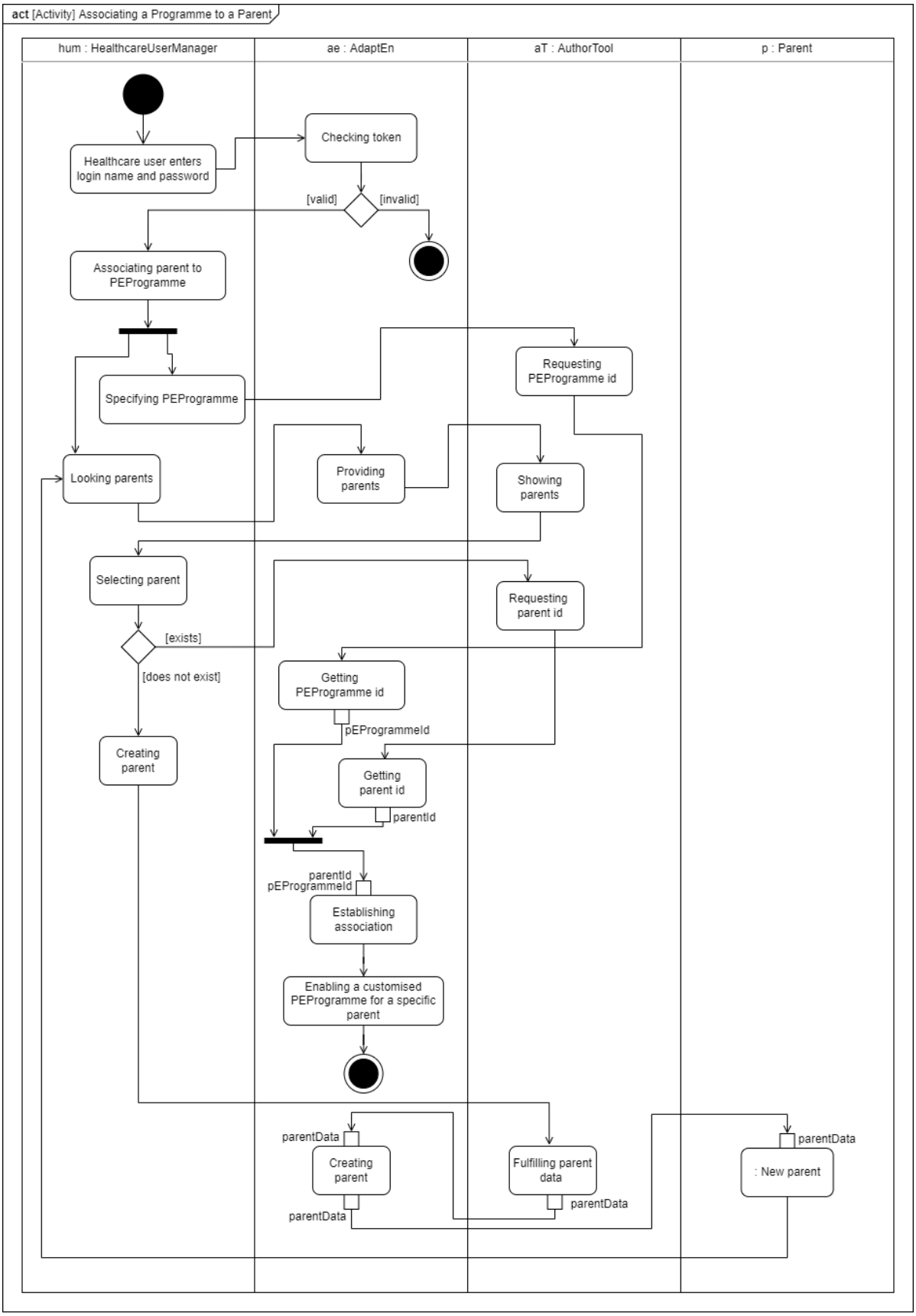


Fig. 4.10 *Activity diagram for associating a psycho-educational programme to a parent.*

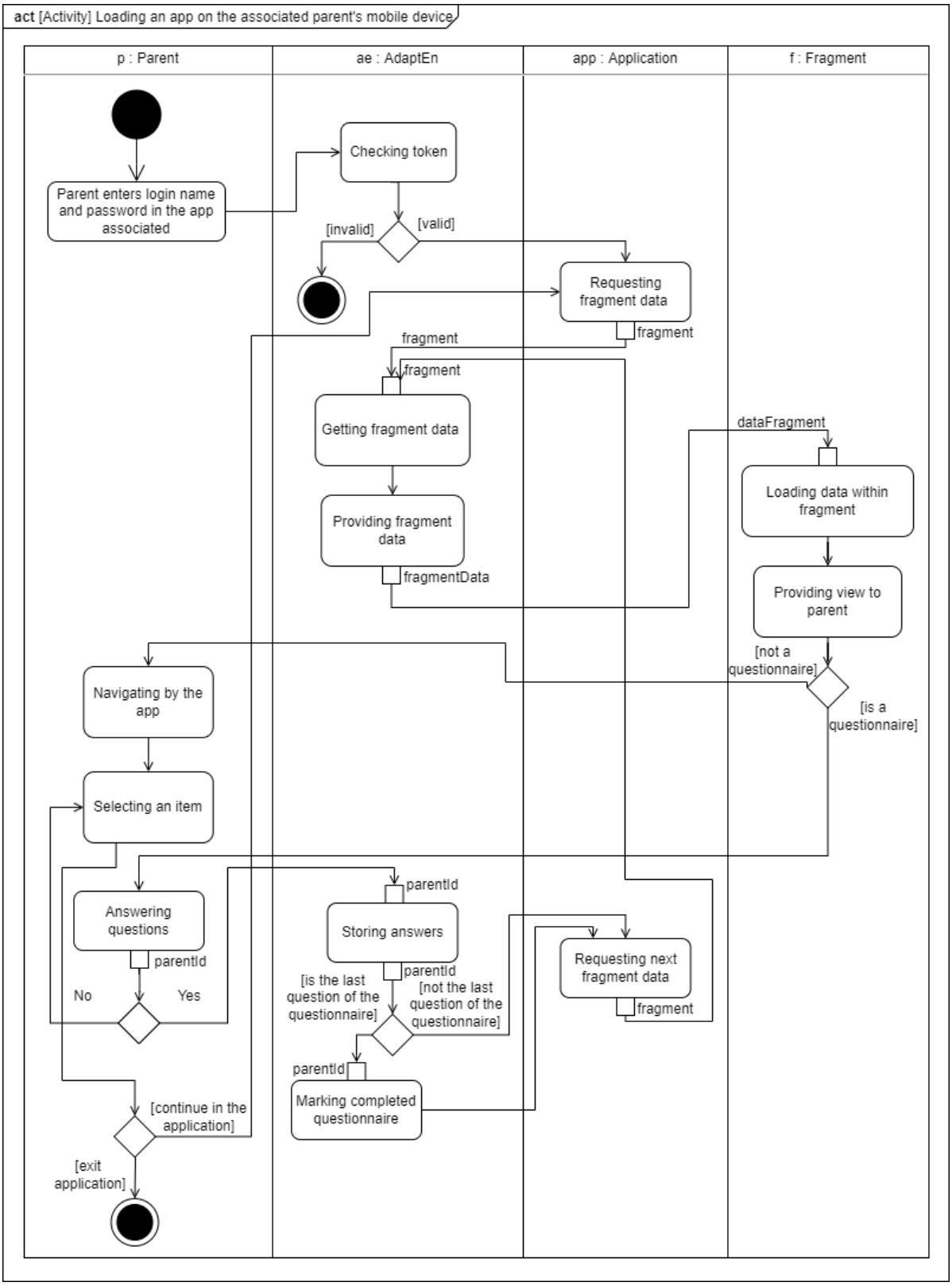


Fig. 4.11 *Activity diagram for loading an app on the associated mobile device.*

4.3.4 Context&Health App

The overall objective of the system focuses on monitoring the emotional state of the children with special needs and their behaviours in response to certain stimulus.

The functional requirements specification of the software system are summarised as follow:

- **FR.1** Provide a tool for designing multi-element integrated programmes. An those elements can be a type of image, music, video or a package. All of them must be labelled. The first three are individual and independent elements but the package is a combined set of the above in order to make it possible to create a special link, relationship, or grouping (e.g., a thematic grouping of a set of nature elements, or images with a blue background that will commonly used in a given program).

- **FR.2** Within a programme, the elements can be displayed sequentially or in parallel, i.e., several at the same time (e.g., picture with music in the background). However, it should be configurable the order and period of execution of each element of the programme (e.g., the time an image must be projected).

- **FR.3** The stimulus may be associated with a previously defined program or may be incorporated by the educator during the development of a work session with the child.

- **FR.4** Programmes must exist to monitor a child. Each child can have associated several programmes.

- **FR.5** Before starting each monitoring session, it is necessary to select the child to work with and the appropriate programme.

- **FR.6** A stimulus board should be provided that includes noise, colour, brightness, movement, vibration, touch, and the possibility to include the whole set. During a monitoring session, the educator should be able to instantly record on the stimulus board the impact that the specific stimulus is having on the child according to his/her perception on a positive, neutral or negative scale. Subsequently, a temporal correlation would be established, in order to make a concrete association between the specific stimulus/s and the impact on the child according to the scale item.

- **FR.7** It is necessary to have stored the profile information of the child. Together with the profile information, personal information (i.e., name, surname, date of birth, and photo) must be stored.

- **FR.8** An initial sensory profile may include hypersensitivity, degree of blindness, motor impairment in lower extremities, etc. This profile can be completed on the basis of information gathered in the following sessions.
- **FR.9** With the system should be possible to manage users (i.e., children), the programmes and their elements, and the stimulus board to monitoring and record the impact produced to the child and perceived by the educator during each session. In parallel, physiological data in a non-intrusive way and environmental data should be collected automatically.
- **FR.10** The physiological data to be considered are acceleration, blood pressure, heart rate, heart rate variability, electrodermal activity, and body temperature.
- **FR.11** The environmental data to be considered are luminosity, noise, temperature, humidity, and atmospheric pressure. An event marker and a time marker should be included to record specific points in time.
- **FR.12** Child profiles, child information, programmes and elements should be added, consulted, edited and deleted.
- **FR.13** Search options must be provided (i.e., user search, program search and program search).

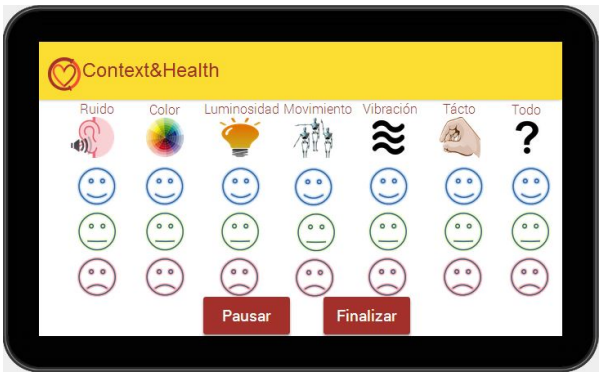
These requirements of the system were defined in collaboration with educators, psychologists and therapists from the centre of integral human, educational and rehabilitative care for children with special needs, who supervised and reviewed the proposed interfaces.

The system proposed to be implemented includes a set of sensors to monitor the physiological variables among which is the E4 wristband (Empatica bracelet) [73]. To monitor environmental variables it was proposed the use of CEnMO App (Section 4.3.2). Moreover, an application with three distinct parts (users, programmes and monitoring) was designed. Figures included in 4.12 show a sample of some of the functionality that this system would include. Its design leads to facilitate the creation of programmes to study and obtain the sensory profile of children with special needs or other specifications whose functionality requires a similar structure.

(a) *Personal user and profile information.*

(b) *Elements that compose a program and their features.*

(c) *Configuration board for defining the sequential or parallel order of the elements.*



(d) *Stimulus board to record at runtime the impact of the elements shown to a user with special needs.*

Fig. 4.12 *Context&Health App interfaces.*

4.4 Systems Functionalities and Technologies

To conclude the chapter, we have drawn up three tables summarising the requirements, characteristics, technological aspects, proposed solutions and studies carried out.

System requirements for each of the cases addressed are included in Table 4.1. Here, it is shown that the requirements of the cases demanded by the domain experts (doctors, psychologists, nurses, midwives and educators) include the collection of information (physiological and environmental) but do not necessarily require the integration of sensors/wearables in the system. It is also relevant that within the system itself it would be possible to store information or create profiles of the patients or users to be monitored. Regarding the variables to be monitored, it is necessary to provide the option for selecting the factors to monitor for each situation, including the possibility of adjusting the sample rate. It is also frequently requested that data can be viewed per session, processed, and stored, as well as that the events or marks can be recorded to review an event a posteriori and that it can be traced. Several applications with specific functionality for different stakesholders are also frequent among the usual requirements.

Table 4.2 comprises the solutions provided on the basis of previous requirements. Most of our system proposals use service oriented architectures with the possibility of specialisation or microservices. The services and microservices have been organised by layers and mainly comprises connection and communication management, storage and processing. Regarding data, we have applied data buffering techniques and storage has been both local and external depending on the case. With regard to technological devices, it is worth mentioning the use of low-cost technologies (Arduino), others from specialised companies, and the system itself integrated with the CEnMo application that we have developed ourselves. The development tasks have been carried out mainly with Android Studio, Angular.sj and Ionic.

The main aspects of the two studies conducted are summarised in Table 4.3. The objectives have been the non-subjective analysis of sleep quality in patients with SLE and the assistance to pregnant women through psycho-educational programmes, respectively. The developed systems have been used in combination with other measuring devices. The second study integrates within the system the questionnaires completed by expert collaborators in the application domain.

Requirements are atomic and can be used repeatedly. In fact, there are strong similarities between the domains studied, which are reflected in the requirements and proposed solutions. These similarities and differences are particularly pronounced when it comes to monitoring systems for data collection with sensors and wearables, and when what is intended to be

implemented is a system to host intervention programmes. However, as shown in the following tables, a combination of the two is in demand, resulting in more complete solutions.

Table 4.1 *System requirements for each case.*

		SMODIAT	Mobile System for Monitoring the Environmental Context	Psycho-educational programme via Mobile Health system (mPOP)	Context&Health App
Devices					
	Sensors aggregation	✔	✖	✖	✖
	Wearables aggregation	✔	✖	✖	✖
	Specific assignment of devices to users	✔	✔	✖	✖
	Sensors already integrated	✖	✔	✖	✖
Psycho-educational programme					
	Programme content management	✖	✖	✔	✔
	Support for the organisation of content	✖	✖	✔	✔
	Audiovisual content management	✖	✖	✔	✔
	Textual content management	✖	✖	✔	✔
	Links management	✖	✖	✔	✔
	Questionnaire management	✖	✖	✔	✔
Data					
	Physiological variables	✔	✖	✖	✔
	Environmental variables	✖	✔	✖	✔
	User personal information	✔	✖	✔	✔
	Search options	✔	✔	✔	✔
	Data per session	✔	✔	✔	✔
	Data preprocessing	✔	✔	✖	✖
	Data storage	✔	✔	✔	✔
	Access to data monitored	✔	✔	✖	✖
	Access to personal information	✔	✔	✔	✔
Monitoring					
	Selection of measures to be recorded	✔	✔	✖	✖
	Sample rate settings	✔	✔	✖	✖
	Real time visualisation data monitored	✔	✖	✖	✖
	Alarm setting	✔	✖	✖	✖
	Event marker	✖	✖	✖	✖
	Notifications	✔	✖	✖	✖
Stakeholders					
	Member of the specialist team	✔	✔	✔	✔
	Family member	✔	✖	✔	✖
	Patient/User to be monitored	✔	✔	✔	✖

Table 4.2 *System design proposals for each case.*

		SMODIAT	Mobile System for Monitoring the Environmental Context	Psycho-educational programme via Mobile Health system (mPOP)	Context&Health App
Services		✔	✖	✔	✔*
Microservices		✖	✔	✔	✔*
Specialisation		✔	✔	✔	✔*
Architecture layers					
	Connection and communication management between devices	✔	✖	✔	✔*
	Data storage	✔	✔	✔	✔*
	Data processing	✔	✔	✔	✔*
	Analysis	✔	✖	✖	✖*
Applications					
	Member of the specialist team	✔	✔	✔	✔*
	Family member	✔	✖	✔	✔*
	Patient/User to be monitored	✔	✔	✔	✔*
Data					
	Data buffering	✔	✖	✔	✔*
	Local storage	✔	✔	✖	✖*
	Remote storage	✔	✖	✔	✔*
Technologies					
	Arduino	✔	✔	✔	✔*
	NexGen Ergonomics	✔	✖	✖	✖*
	MotionWatch 8 (Actigraphy wristband)	✖	✔	✖	✖*
	Empatica e4 (physiological signals)	✖	✖	✔	✔*
	CEnMo	✖	✖	✖	✔*
Development frameworks used					
	Android Studio	✔	✔	✖	✖*
	Angular.js + Ionic	✖	✖	✔	✖*

(*) means that the system was designed but not implemented.

Table 4.3 *Studies conducted.*

Study		Objective analysis of Sleep Quality on SLE patients	Assisting pregnant women through a psycho-educational programme
Systems used			
	Mobile System for Monitoring the Environmental Context (CEnMo)	✔	-
	mPOP	-	✔
Devices			
	Own device	✖	✖
	Device with built-in sensors (CEnMo)	✔	✔
	MotionWatch 8 (Actigraphy wristband)	✔	✖
	Empatica (E4 wristband)	✖	✔
Questionnaires designed by our experts		✖	✔
Standardised questionnaires		✔	✖

Part III

ASTREA Framework: Design and Modelling

Chapter 5

ASTREA Framework

Chapter Abstract

Nowadays, modern monitoring systems are on the rise in our society because technological advances are supporting monitoring, remote control, and data gathering everywhere and at any time. However, their complexity is also increasing due to internal and external factors such as contextual conditions, user needs, changes in the functionality, services availability, heterogeneity of devices (sensors and wearables), and resources availability. All of these generate uncertainty and situations that are difficult to predict at design time prior to system deployment. To our knowledge, there is no end-to-end solution that supports making changes to system functionality and deploying them at runtime in a dynamic environment using mobile elements and, crucially, maintaining localised scalability as the property of a system not to send information beyond where it makes sense to send it by avoiding interaction between distant entities [207].

ASTREA framework has been created with the aim of providing solutions on three basic pillars: 1) to enhance the reuse of software, in particular, design and development of monitoring systems; 2) to support autonomous deployment, and propagation of adaptations and upgrades to system functionality at runtime; and 3) data gathering. The latter two have been designed to operate in dynamic networks and with mobile elements. ASTREA also includes a visual editing tool for domain-specific software developers to speed up the prototyping of systems by abstracting them from low-level technical issues and providing them with a set of sensors, wearables, and functionalities that they can reuse by freeing them from programming. In this way, developers of domain-specific software can be focused on system design. With the aim of illustrate and animate the capabilities of ASTREA framework, this chapter also includes a case of study and an implementation to demonstrate the feasibility of the proposal deployment in a test environment.

Chapter Contents

5.1 Introduction

This chapter presents ASTREA framework as an integral solution for common but still challenging issues of monitoring systems. On the one hand, monitoring systems frequently require adaptations and upgrades after deployment. On the other hand, monitoring system requirements must be fulfilled for the collection of measurements (physiological and environmental data), maintaining or improving the performance of the system.

There are many scenarios in which monitoring systems can be deployed, where changes are frequent due to context information (i.e., user context, computing system context, physical context and temporal context) and for which customised or adaptive services must be provided [263, 64]. WSNs or BSNs should be attached in a non-intrusive manner to users (patients). These users and others (e.g., caregivers) could move acting as mobile data carriers and node routers supporting multi-hop connections. Within ASTREA, we take advantage of these mobile elements in order to promote the transmission of data (e.g., specification of a monitoring system, parameter specification, *code mobility*, *corrective adaptation*, *adaptive adaptation*, *extending adaptation*, *perfective adaptation*, *compositional adaptation*, at different levels and dimensions, etc.) [135]. This provides a way for systems to be modified and adapted as many times as necessary, easily and quickly, with minimal human intervention, and at runtime, without the need to reinstall applications. In addition, we also address another common goal of data gathering by analysing the advantages of incorporating in-network preprocessing and mobile elements (or devices) that act as carriers.

In this chapter, firstly, the usefulness of the proposal is illustrated in a motivating scenario. Next, it can be found the specific objectives of ASTREA framework, and subsequently the system model that frames ASTREA. The idea of the *case* concept, one of the main foundations of the proposal, is introduced. Subsequently, ASTREA architecture is presented, its main components are detailed, as well as the developed mechanisms to support autonomous deployment, and propagation of adaptations and upgrades of system at runtime, and data gathering.

The deployment, and propagation of adaptations and upgrades mechanism allows these systems to be put in place with virtually no human intervention to install or configure the devices, they just need to be located within the scenario and switched on. Furthermore, systems need to be reconfigurable in order to cover the needs of monitored users and for better system performance or optimisation from the introduction of in-network preprocessing operations (increasing success rate of meaningful information, minimising bandwidth and saving energy). This reconfiguration must be carried out autonomously (i.e., self-configuration and self-optimisation) on the part of the system. In addition, ASTREA Case Editor (ASTREACE)

tool is shown, from which the specification of the *cases* is created or modified by a case designer (i.e., domain-specific software developers or domain-specific users). With ASTREACE tool, the case designer can select, drag and drop the devices to be used (sensors and wearables) and actions as well as to set certain configuration parameters to provide functionality to the monitoring system (i.e., *case*). Actions involve in-network processing, storing, and self-adaptive specifications.

Once cases are created, they must be deployed, which is handled by the *deployment* mechanism. The *case subversion* service generates a version of the case specification for each of the nodes involved in the system which are sensors and wearables (sources), mobile elements that are data carriers (intermediaries), and data sinks (destinations). In this way, each node receives only what it needs and network infrastructure can be shared as environmental sensors can collect information that can be useful for different cases. Notwithstanding, data transmission must be valid regardless of the purpose of the system (requirements or actions). The following is a detailed case study comprising the complete process for creating a case, deploying it, and gathering the data with ASTREA. It also briefly outlines the adaptation plans incorporated within the framework, as well as the most relevant information about the *service repository*. Finally, a sample of the implementation is included which is used in a test deployment to show the dynamic composition of the services and microservices in a distributed environment.

The most relevant parts of ASTREA will be modelled according to the 4+1 View Model using Systems Modeling Language (SysML) that supports the modeling of specification, analysis, design, verification and validation phases [232]. The 4+1 View Model organises the description of the architecture in five concurrent views: 1) Logical View; 2) Process View; 3) Physical View; 4) Development View; and 5) Use Case View. In this way, the organisation of the architecture represents the structural elements, the behaviour and composition of both to form subsystems and to meet stakeholder needs [160]. Here, structural diagrams define the static architecture and behavioral diagrams represent the dynamic part [160].

5.2 Motivating Scenario

In this section, to animate the capabilities of the system and interplay between the primary components, we will now consider a nursing home scenario.

Within the nursing home, we assume there are several residents which require to be monitored. The monitoring process primarily addresses *what* to measure, *where*, *how* and *when*. With regard to *what* to measure, the proposal for *data gathering* involves:

1. physiological variables useful in determining the condition of the residents;
2. environmental variables useful in determining their surrounding conditions; and
3. residents location, useful in determining automatically where she or he is, and the possible implications of being there.

Furthermore, the area *where* the residents are moving can be relevant and it could encompass a wide area, taking into account different spaces included those which are visited only for short periods of time, either (1) *indoor*, and (2) *outdoor*. In addition, with respect to *how*, it is also worth noting, that it is possible to find different degrees of residents' mobility within a space. Finally, *when*, it should be managed directly by the monitoring system itself considering that monitoring should be continuous as long as necessary to meet system requirements. All measures should be gathered objectively and without human intervention from an infrastructure of heterogeneous sensors, wearables and other devices (e.g., nodes with higher computational capabilities) that compose the dynamic distributed wireless networks.

In the nursing home, changes are frequent. Residents are not always the same, as there are new residents join but also leaving. In this regard, residents' needs are not equal and it is important that systems can be adapted or customised to serve them optimally.

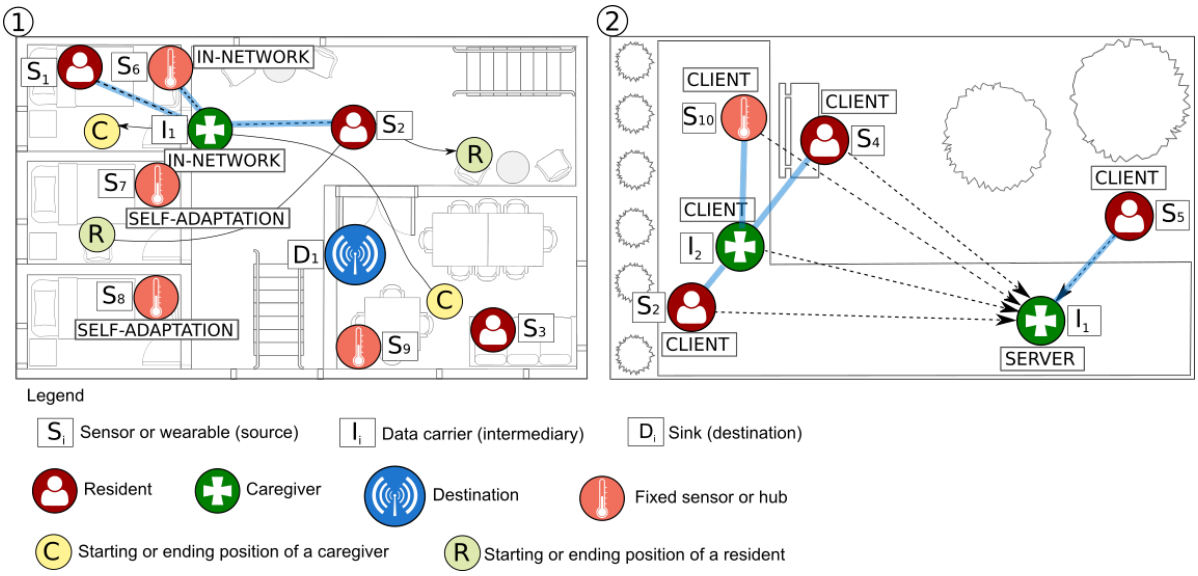


Fig. 5.1 *(1) Indoor cases (on the left), arrows reflect the data transmission for cases deployment, propagation of adaptations and upgrades. (2) Outdoor cases (on the right), arrows reflect the data transmission for data gathering.*

As example, in Figure 5.1 can be seen that there are five residents (S_1, S_2, S_3, S_4 and S_5) with BSNs, two caregivers (I_1 and I_2), and one destination (i.e., sink node). Each resident is

monitored by a custom-designed monitoring system (i.e., case). Furthermore, there are five hub of environmental sensors (S_6, S_7, S_8, S_9 and S_{10}) which share the same instance of a case being an instance a replica of a same case. Residents and caregivers can walk, which is represented by R and C points respectively. In particular, resident S_2 is initially in his/her room but then walks to the chairs in the hall while the caregiver I_1 who is initially in the living room (C) goes to visit resident S_1.

ASTREA makes possible to specify and manage the case for each resident in a transparent manner. It is shown, in the Figure 5.1 (on the left) that I_1, who is moving, has in its reach range S_1, S_6, and S_2. I_1 carries the case specification for each of these nodes and will send them to them. In parallel, sensors, wearables, and WSNs are already collecting data but their configuration, in-network operations and how to gather the data will be set once they receive its case specification.

The image on the right (Figure 5.1) represents that I_2 has within its reach range to S_2 (this resident has gone outside for a walk), S_4 and S_{10}. However, I_1 is a better host of the data than I_2 because its computational capabilities at that time are better (e.g., more remaining storage or battery). Therefore, S_2, S_4 and S_{10} will send the data to I_1 via multi-hop connection. Finally, I_1 will retransmit the data it has collected to a destination when the latter is in range.

5.3 Specific Objectives

We present the ASTREA framework as a generic solution to:

1. speeding up the prototyping of monitoring systems;
2. autonomous deployment of monitoring systems in mobile WSNs with dynamic network topologies;
3. modifying the behaviour of already deployed monitoring systems at runtime;
4. introducing in-network preprocessing operations in order to optimise WSN performance in terms of increasing success rate of meaningful information, and minimising bandwidth and energy consumption.

5.4 System Model

This section provides an overview of ASTREA framework using a block diagram in SysML (Figure 5.2) as well as the most relevant assumptions.

ASTREA framework comprises three main entities:

- *Case Editor (ASTREACE)* tool is an editing tool that supports the creation and later modification of *cases* (detailed in Section 5.10). A *case* is the specification of a monitoring system in ASTREA.
- *Monitoring System (ASTREAMS)* is the core of ASTREA framework which supports the first case deployments and subsequent case modifications (i.e., adaptations and upgrades) as well as data gathering. This transmission of data is conducted by *autonomous deployment* and *reconfiguration at runtime* (detailed in Section 5.9.1), and *data gathering* (detailed in Section 5.9.2) mechanisms. Both mechanisms are managed independently.
- *Repository* contains reusable software entities, in particular, services and microservices to compose the systems developed with ASTREA framework.

ASTREA framework can involve several stakeholders (i.e., *doctors*, *nurses*, *psychologists*, *midwives*, etc.) which can collaborate with the software engineers and case designers. The software engineers are the ones who enrich the *service repository* and integrate the hardware devices (i.e., *nodes* to collect *measurements* and carry out *actions*) within ASTREA, while the case designers are the ones who create the cases (i.e., monitoring systems) from the elements available at ASTREACE, and microservices provided in ASTREA and previously created by the software engineers. Case designers usually have extensive knowledge of the application domain.

Data collection in ASTREA is envisaged as the monitoring of *physiological* variables (i.e., measurements) of users, and *contextual* variables (*environment* or *location*). These data are collected with sensors or wearables objectively without human intervention. What variables should the monitoring system include, must be specified within the *case* specification. This *case* specification should also include what *actions* must be applied over these variables. The measurements monitoring and actions integrate the system's operations encapsulated within ASTREA's services and microservices. The *case* specification, and services and microservices associated will be deployed within the *nodes* of a dynamic network. Further details on the types of nodes that compose the network infrastructure in ASTREA can be found at Subsection 5.7.1.

In addition, the following underlying considerations have been taken into account in ASTREA design and development:

- Each *case specification* defines an independent monitoring system but in a scenario can coexist multiple *cases*.

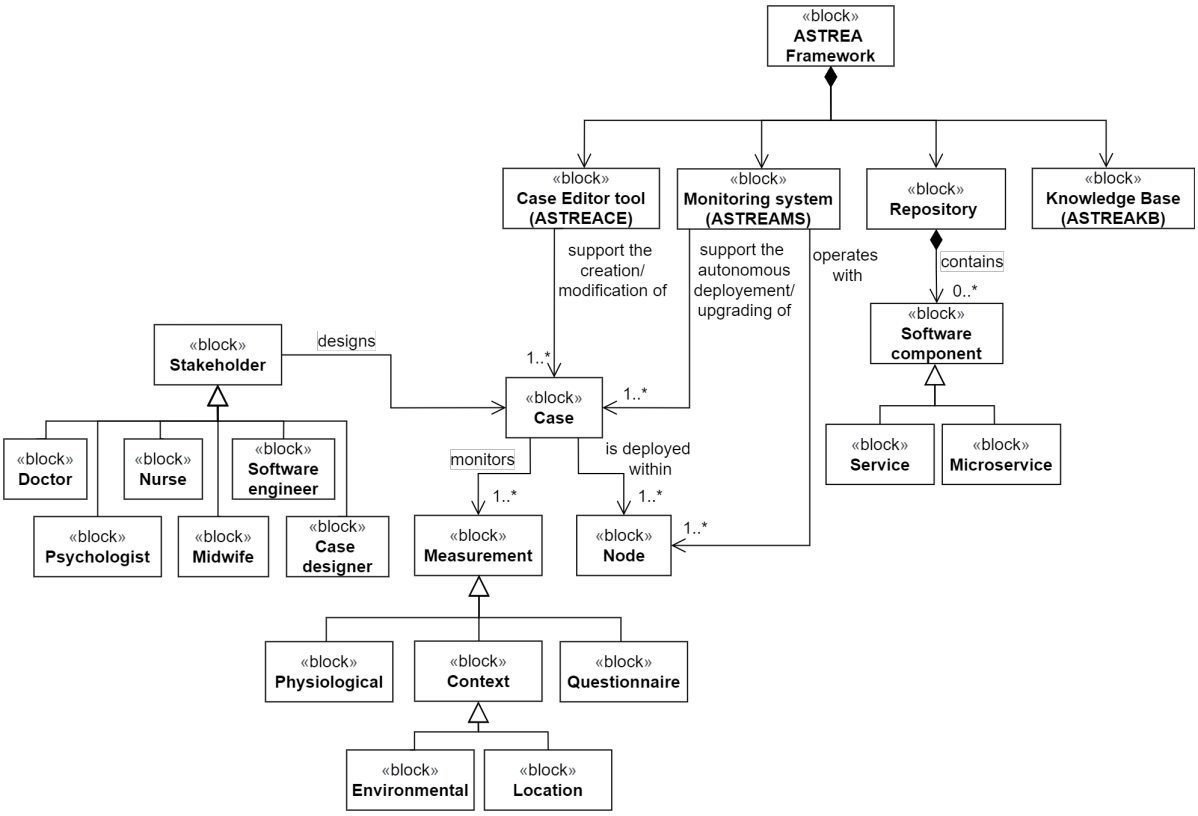


Fig. 5.2 *ASTREA framework block diagram in SysML.*

- Devices (i.e., mainly sensors and wearables) can be added and removed from the *cases* at runtime.
- In ASTREA network infrastructure one device or a set of them are considered as nodes.
- The network infrastructure can be shared. This means that a node can be used at the same time for several *cases*.
- Devices that compose the monitoring systems are heterogeneous.
- Actions are what each system does in a granular level.
- Actions can be reused by several *cases*.
- Devices and actions are selected by a *case designer*.
- *Adaptation* mechanisms can be introduced by a *case designer*.
- *Self-adaptation* mechanisms can be defined by a *case designer* and executed autonomously.
- The *adaptation* and *self-adaptation* mechanisms can be executed in parallel in multiple *cases*.
- Communications in dynamic networks are unreliable.
- Distance between nodes can cause channel fading.
- Buffer size is limited and congestion within the network is possible.
- Links between nodes could not be bidirectional. Nodes can have different range for connection and consequently for data transmission. Therefore, a node could send data but the receiver could not have the sender within its achievable range.
- Multi-hop communication is managed by the routing protocol.
- ASTREA framework follows an *opportunistic* approach where there are no restrictions about *delay-tolerant*.
- Nodes (network devices) are not aware of the existence of other nodes (with the same or different role) in the scenario.

5.5 Adaptation Plans

ASTREA supports three types of adaptation:

- Adaptation in already deployed monitoring systems. Adaptations are made on the basis of the modifications that can be introduced to the systems and applied via the mechanism for autonomous *deployment, and propagation of adaptations and upgrades*. These adaptations are linked to upgrades or the addition of new functionality at runtime and without recompiling the already deployed system. A case designer can make modifications to his/her initial specification through the ASTREACE visual editing tool and it can also incorporate new devices whether it want to take more measurements and/or incorporate additional actions. It also can remove both, devices and actions.
- In the *data gathering* mechanism, adaptation can be found at the level of data prioritisation, data buffering, or election of the target node by data forwarding policy.
- The case editor can also include in the case specification itself, self-adaptation mechanisms in order to improve system performance in terms of energy savings. The monitoring system, once deployed, will be performing checks to identify when the conditions defined are satisfied to perform adaptation autonomously.
- Microservices that make up the monitoring systems can be downloaded and transmitted over the network because code mobility is feasible in ASTREA.

5.6 Case Concept Formalisation

A specific monitoring system will be design on the basis of the domain knowledge. Furthermore to the domain knowledge, the design and infrastructure (devices or nodes) used for other previous systems could be an useful source of knowledge or even could be reused. At the end, this information is part of a case specification and can be understand as a dual approach combining top-down and bottom-up approaches [21]. The *top-down* is about approaching the problem from a general perspective until the lowest level is reached and the *bottom-up* refers to the process that goes from the simplest level to the global problem [161]. In particular, the *top-down* divides more complex cases into simplified cases, and *bottom-up* combines the simplest cases to compose more complex cases. Complementary, Crespi et al. [60] consider that *top-down* assumes that each component has a global knowledge of the system, while

bottom-up design starts with the specification of the individual requirements and capabilities of each component.

Pearce et al. [183] notices that the design of *"complex artifacts"* involves: *"abstract design concepts", "skeletal design plans", "selecting specific systems and components", or "accepting design solutions" inter alia*. Dooley [69], for his part, considers a case to be an event-driven description of a real-life activity or problem.

ASTREA is based on the *case* [183, 21] concept to cover monitoring system specification, which includes the use of (1) hardware devices (e.g., smartphones, sensors, wearables, micro-controllers, motes, among others); (2) actions such as data gathering (both physiological and environmental measurements), data preprocessing, data storage; and (3) devices state checking to improve performance in terms of network overhead and power savings. The *case* structure results from the study and development of several monitoring systems (as described in the previous chapter, Chapter 4) focused on sleep disorders, women at risk during pregnancy, monitoring people's daily routine, assessing the environmental quality, inter alia. Despite of being systems with a priori different functional requirements, they all share key actions, functional flows, and hardware devices to monitor the health state of patients.

In ASTREA framework, the case specification is a description of the monitoring system to be deployed in a mobile and dynamic WSN. The requirements of a case are defined by one or more actions specified by the case designer. Figure 5.3 shows the case diagram block in SysML which is part of the Logical View. The Logical View comprises the structural part of the specification of the functional system requirements at different levels [160, 133]. The specification includes which actions a case specifically includes and how they are interrelated. In particular, the type of actions can be:

- *Device* actions to monitor environmental (i.e., environmental sensors) and physiological variables (i.e., sensors or wearables). Device actions include configuration setting parameters. If the device does not allow to be configured, it will be set at software level.
- *Preprocessing* actions to perform in-network operations related with a measurement or the device which takes the measurement.
- *Storage* actions to store information (raw data or preprocessed data) persistently.

All these options are selected and configured by a case designer.

Figure 5.4 shows an example of cases scheme with three cases. $Case_1$ includes three actions ($Action_1$, $Action_2$ and $Action_3$) that are included in two requirements (which would define an objective from a more general perspective). $Case_2$ and $Case_3$ include two requirements,

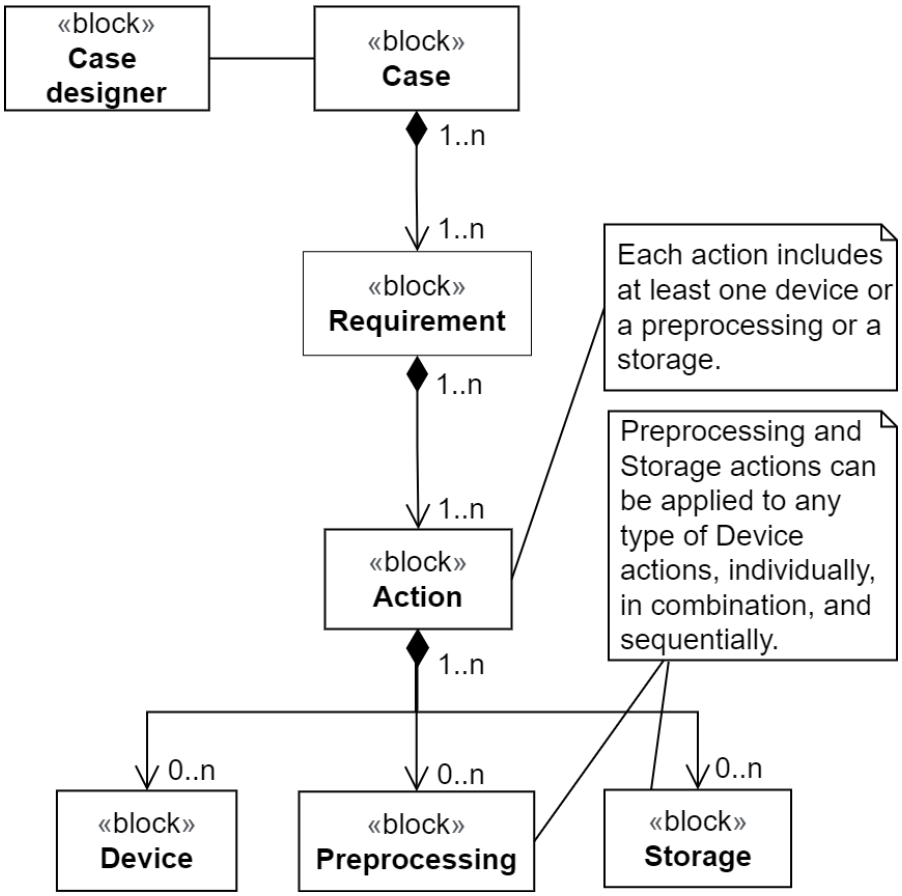


Fig. 5.3 *Case block diagram in SysML.*

with five associated actions ($Action_2$, $Action_4$, $Action_5$, $Action_6$ and $Action_7$) and two actions ($Action_1$ and $Action_2$) respectively. At ASTREA, reuse is possible at different levels. On the one hand, $Case_2$ reuses $Action_2$ and $Preprocessing_3$ already defined within $Case_1$. On the other hand, within $Case_3$ specification, the requirements ($Requirement_1$ and $Requirement_3$) already defined within $Case_1$ and $Case_2$ are included.

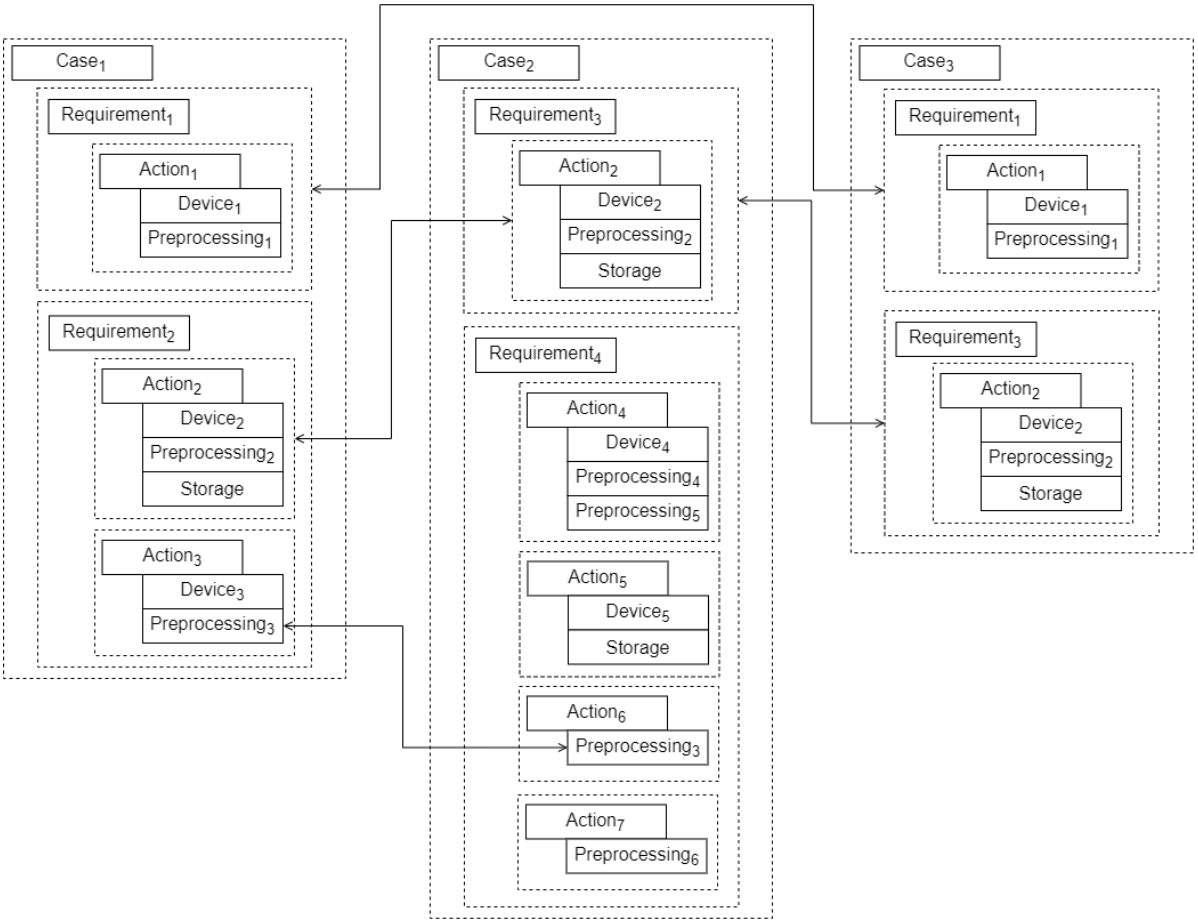


Fig. 5.4 *Example of cases scheme.*

Once the case designer finalises the specification, ASTREA transforms the case specification to a JSON file and with the deployment mechanism (Section 5.9.1), the monitoring system will be deployed within the mobile and dynamic WSN. Latter, the case designer can make adaptations and modifications to the case (i.e., the monitoring system) already deployed.

Finally, it should be noted that these monitoring systems (i.e., cases) are not intended to control critical patients situations but monitor patients in their daily lives.

5.7 Architectural Design

ASTREA framework architecture (Figure 5.5) comprises:

1. a *distributed microservice architecture*, called ASTREA Monitoring System (ASTREAMS), intended to provide support for the two following mechanisms:
   (a) cases (i.e., monitoring systems) deployment, and adaptations and upgrades at runtime, and
   (b) data gathering in mobile wireless network infrastructures;
2. a *service repository* which includes all the microservices that can be deployed on a node according to the particular case specification; and
3. a *visual editing tool*, namely ASTREA Case Editor (ASTREACE), to create the specification of the *cases* by a case designer as well as to adapt and upgrade them at runtime. In particular, when the case editor injects adaptations and upgrades via ASTREACE, this produces changes into the case specification of the monitoring system.

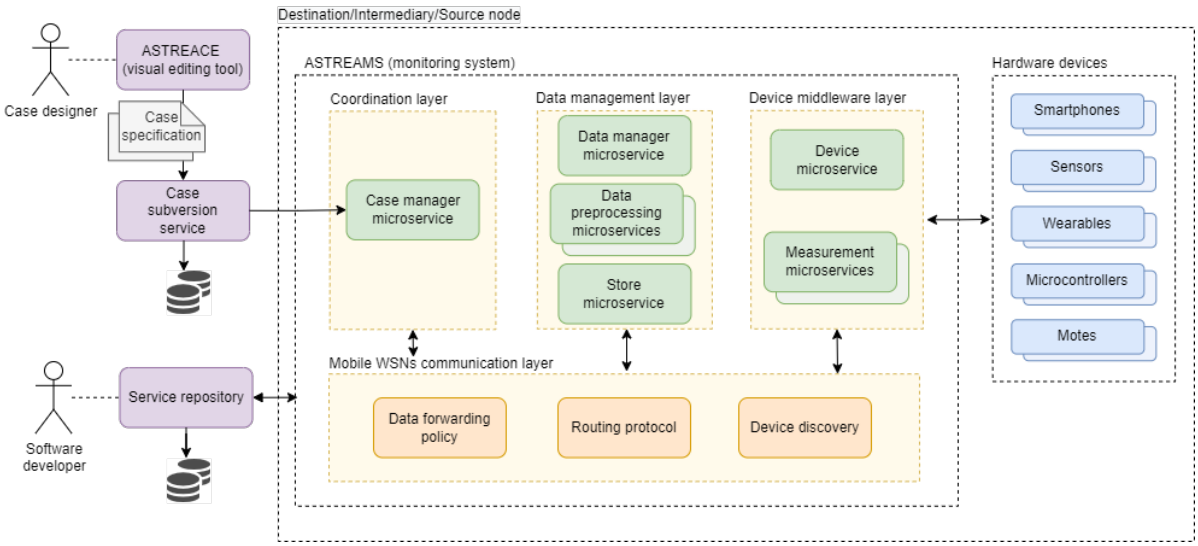


Fig. 5.5 *ASTREA architecture.*

5.7.1 Characterisation of ASTREA Nodes

The nodes that can be found within the ASTREA framework are characterised by roles: *source*, *intermediary* and *destination*. Each role possesses the specific capabilities and properties

summarised in Table 5.1. Moreover, in Figure 5.6, it is shown the specialisation of nodes in terms of mobility.

Source nodes can be or not mobile, with regard to computational capabilities, it should be noted that they may be limited in comparison with *destination* and *intermediary* nodes. *Intermediary* nodes mobile, and they act as a link from *destination* nodes to *sources*, both to propagate the cases specification, adaptations and upgrades introduced by the case designers at runtime and to carry the data gathered. Storage capacity, power consumption, and communication range of *intermediary* nodes are limited and lower than the *destination* nodes but greater than the *source* nodes. Finally, *destination* nodes, which are fixed, have high computational capabilities and a continuous power supply, and act as gateways to the WSNs for the data gathering process.

It should be mention that *intermediary* nodes could posses similar capabilities to *destination* nodes. However, in ASTREA, this characterisation has been made because mobile devices generally tend to possess less computing capabilities than fixed devices, as for example, the latter may be connected to a power supply and have a wired network connection.

Table 5.1 *Features and responsibilities of the nodes grouped by role.*

	Data monitoring				Processes		Microservices							
	Physiological	Environmental	Device resources	Mobility capabilities	Deployment, and adaptations and upgrades	Data gathering	Case manager	Data manager	Data preprocessing	Storage	Device	Measurement	Data forwarding policy	Routing protocol
Source	●	●	●	◓	○	●	●	◓	◓	◓	●	●	●	●
Intermediary	○	○	●	●	●	●	●	●	●	●	●	◓	●	●
Destination	○	○	○	○	●	○	●	○	●	●	◓	○	●	●

○ Not supported
◓ Optional
● Supported

5.7.2 ASTREAMS Architecture

ASTREAMS architecture supports the autonomous *deployment, and propagation of adaptations and upgrades* mechanism and *data gathering* mechanism. It involves a set of microservices grouped into four layers: (1) *coordination* layer; (2) *data management* layer; (3) *device middleware* layer; and (4) *mobile WSNs communication* layer. These microservices are deployed and distributed in the nodes that compose the system, according to their role, capabilities and requirements of the monitoring system (Table 5.1).

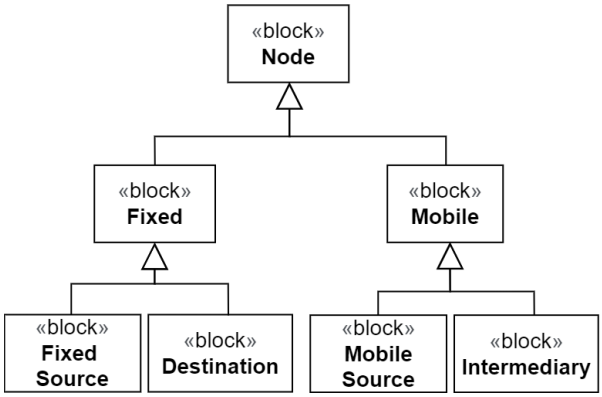


Fig. 5.6 *Node block diagram in SysML.*

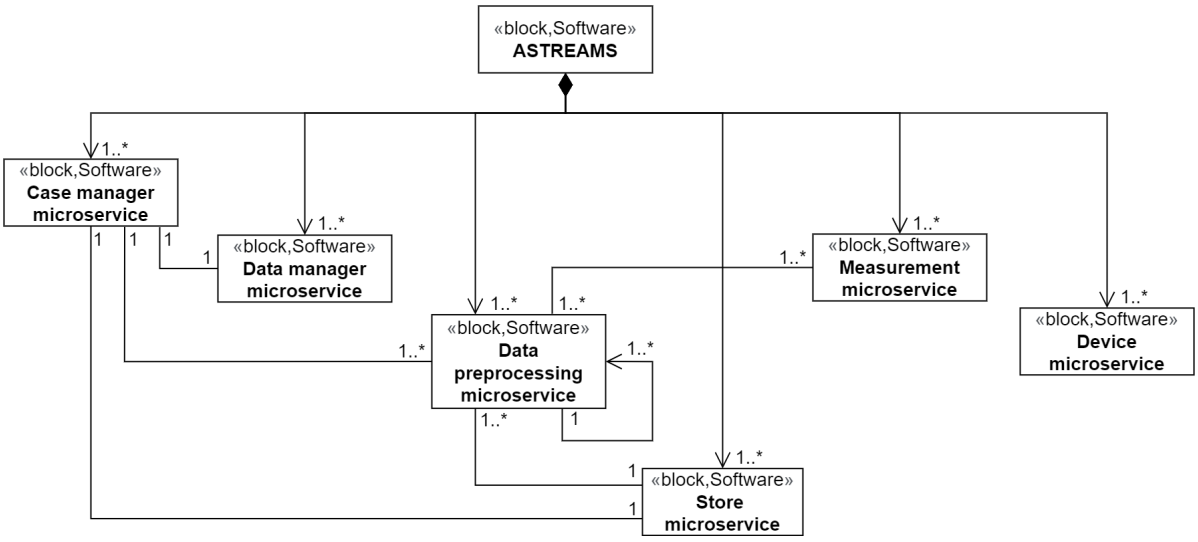


Fig. 5.7 *ASTREAMS software architecture block diagram in SysML.*

- The *coordination* layer includes the *case manager* microservice which is deployed within each node that is part of the network. If a node does not have sufficient computational capabilities to host it (e.g., source nodes computational capabilities could be highly limited), then the node must be associated to another unit with sufficient computational capabilities that includes a *case manager*. The *case manager* microservice is responsible for coordinating the rest of the microservices to comply with the case specifications, both in the deployment as well as in propagation of adaptations and upgrades of the system.
- The *data management* layer involves *data manager*, *data preprocessing* and *storage* microservices.
  - *Data manager* microservice determines what data should be maintained when the remaining storage capacity is insufficient. Therefore, if the storage capacity of a node reaches its limit, the *data manager* checks data priority and retains those with the highest priority. Whether all data have the same priority, the *data manager* checks the timestamp and prioritises the most recent. Destination nodes do not require a *data manager* since they are assumed to have a large storage capacity, which can be treated as unlimited. The *data manager* is also responsible for formatting the data for transmission to other nodes.
  - The *data preprocessing* microservices of ASTREA encapsulate relational, logical, mathematical and aggregation operations on the data gathered. A modular approach is followed which allows to the case designer the reuse and combination of operations (i.e., type of action) enabling complex data processing.
  - The *storage* microservice manages local data storage on the device. It provides operations to store (i.e., type of action), extract and remove the data gathered and implicitly could support data management for *data preprocessing*. However, source nodes may not have sufficient computational capacity to deploy it but it could be deployed within an associated unit or intermediary or destination nodes.

  Several *data preprocessing* microservices can be deployed within the same node, from among those available in ASTREA, if multiple preprocessing operations are required to conform the case specification, but only one storage instance of *storage* microservice can be deployed within a node in order to maintain integrity and data consistency.
- The *device middleware* layer homogenises and provides a common interface for the hardware devices included in ASTREA which supports the gathering of measurements. This layer includes the following microservices:

  - *device* microservice focuses on hardware device management (i.e., hibernate or activate the devices), state monitoring (remaining battery and available storage capacity), and optionally, set reporting frequency; and
  - *measurement* microservices gather and homogenise data from the hardware devices and transmit it to *data preprocessing* or *storage* microservices of the *data management* layer, unless the storage capacity limit has been reached, then the data will be sent to the *data manager* microservice.

  The relationship between these microservices which support the ASTREAMS functionality is shown in Figure 5.7.

  A hardware device can monitor single or multiple measurement. ASTREA requires to deploy in source nodes one instance of *measurement* microservice per measurement to be monitored. Intermediary nodes could also host *measurement* microservices, whether in a particular case it is of interest for the case designer that these nodes gather data. Moreover, a same node can be composed of different *hardware devices* to monitor several measurements. To monitor remaining battery and available storage capacity that are hardware dependent, a single instance of a *device* microservice is required in source and intermediary nodes. This is not required on destination nodes.

- *Mobile WSNs communication* layer involves *data forwarding policies* that supports the data transmission flow within the *data gathering* mechanism. It also includes *routing protocol* and *device discovery* which are necessary in both autonomous *deployment, and propagation of adaptations and upgrades* of system at runtime and *data gathering* mechanisms.

5.8 Case Subversion Service

Case subversion service handles the *case* specifications. It receives a JSON file from which it generates several JSON files, one for each of the source nodes involved within the *case*, and another for all the intermediaries if any and destinations. Figure 5.8 shows an activity diagram (which corresponds to Process View within 4+1 View Model) for generating the case subversion files for a dynamic network in SysML. The *case subversion* service checks inter alia whether the specified actions require the combination of several measurements, whether such measurements are collected by the same source node, and whether the source node has sufficient capabilities to perform each of the specified actions, otherwise that action must be

performed at a higher level node in the hierarchy (intermediary or destination) and therefore this part of the specification is added to the subversion of the latter nodes. However, intermediary nodes that are not known a priori can be incorporated into the system dynamically at some point in time.

When there is a change in a *case*, what takes place is an adaptation or upgrade of the *case* and another version of the *case* is generated. Figure 5.8 also includes adaptation and upgrade considerations. The *case subversion* service compares the current version of the running system with the newest modified version. Subsequently, the *case subversion* generates as many subversions as source nodes have been excluded in the most recent specification, in particular, one JSON file for each node. These JSON files also contain the actions that must be disabled.

In ASTREA, the case version is generated by the case designer with the visual editor (ASTREACE) (Section 5.10). However, it is worth mentioning that both, the case subversion service and ASTREACE are fully functional independent by themselves.

5.9 System Operation

The operation of the system comprises the mechanisms developed at ASTREA that support dynamic adaptations and upgrades. The dynamic behaviours of the systems are represented as activity diagrams in SysML and these correspond to the Process View within 4+1 View Model.

5.9.1 Deployment, and Propagation of Adaptations and Upgrades Mechanism

The *deployment, and propagation of adaptations and upgrades* mechanism consists of deploying the monitoring system in mobile WSNs previously defined by the case designer. This specification is the *case* (Section 5.6) itself and is in a JSON-formatted file.

Whether a system is modified once it has been deployed, a new subversion of the case is created and must also be deployed, which will adapt or upgrade the currently running system. Therefore, the mechanism involves the transmission of the original case version (the set of *subversions*) and associated *microservices* that make up the functionality of the monitoring system. The following assumptions and considerations are made:

- Data transmission flow originates at the destination nodes. All destination nodes receive a notification from the *case subversion* to start the *deployment, and propagation of adaptations and upgrades* mechanism.

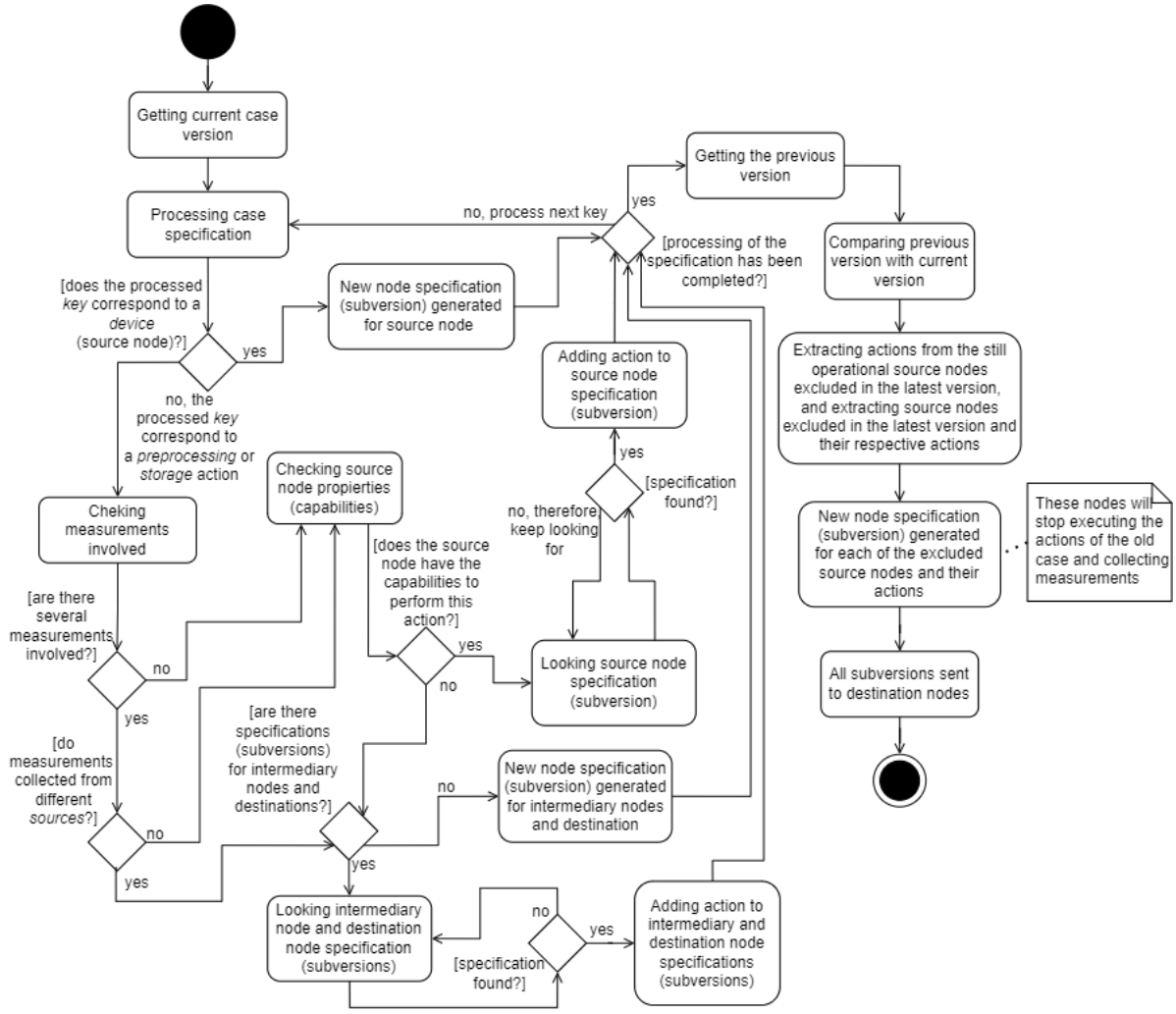


Fig. 5.8 *Activity diagram for generating the case subversion files for a dynamic network in SysML.*

- Destination nodes can receive several subversions that correspond to a case specification. Each subversion includes only the part of the case specification that affects to each specific source node, and to any intermediary or destination nodes.
- Destination nodes request the microservices to cover the specification of each subversion to the *service repository* and store them when they receive them.
- Destination nodes first send each subversion together with the microservices associated to each reachable source node in order not to propagate unnecessary data to the intermediary nodes. Subsequently, they send the subversions of the source nodes that are not directly reachable and their microservices associated to the intermediary nodes which should be reachable at some point. They also send the subversions that must be deployed within intermediary nodes and their associated microservices.
- Source nodes are not transmitters. They can not act as routers in multihop connections.
- Node mobility is non-deterministic, and intermediary nodes and mobile sources could eventually reach a destination node for an indeterminate time. Intermediary nodes and mobile sources may eventually reach into a common range.
- After each transmission, the receiver sends back an acknowledgment message (ACK) confirming when it has received data. The initial sender registers that the receiver node already has its case specification updated together with the microservices associated with it. Therefore, the same data will not be retransmitted again.
- The IDs of the nodes already upgraded are registered in the log file together with the specific subversion of the case. If the node has not been completely upgraded, but only partially upgraded, its ID is not registered in order to synchronise the log files to check what the node is missing to be complete.
- When intermediary and destination nodes have registered the ACK of all nodes to which they can propagate the subversions and associated microservices, they delete these subversions and not needed microservices associated to free up space.

The entities that participate when executing this mechanism, how they interact with each other and the execution flow, is detailed in Figure 5.9.

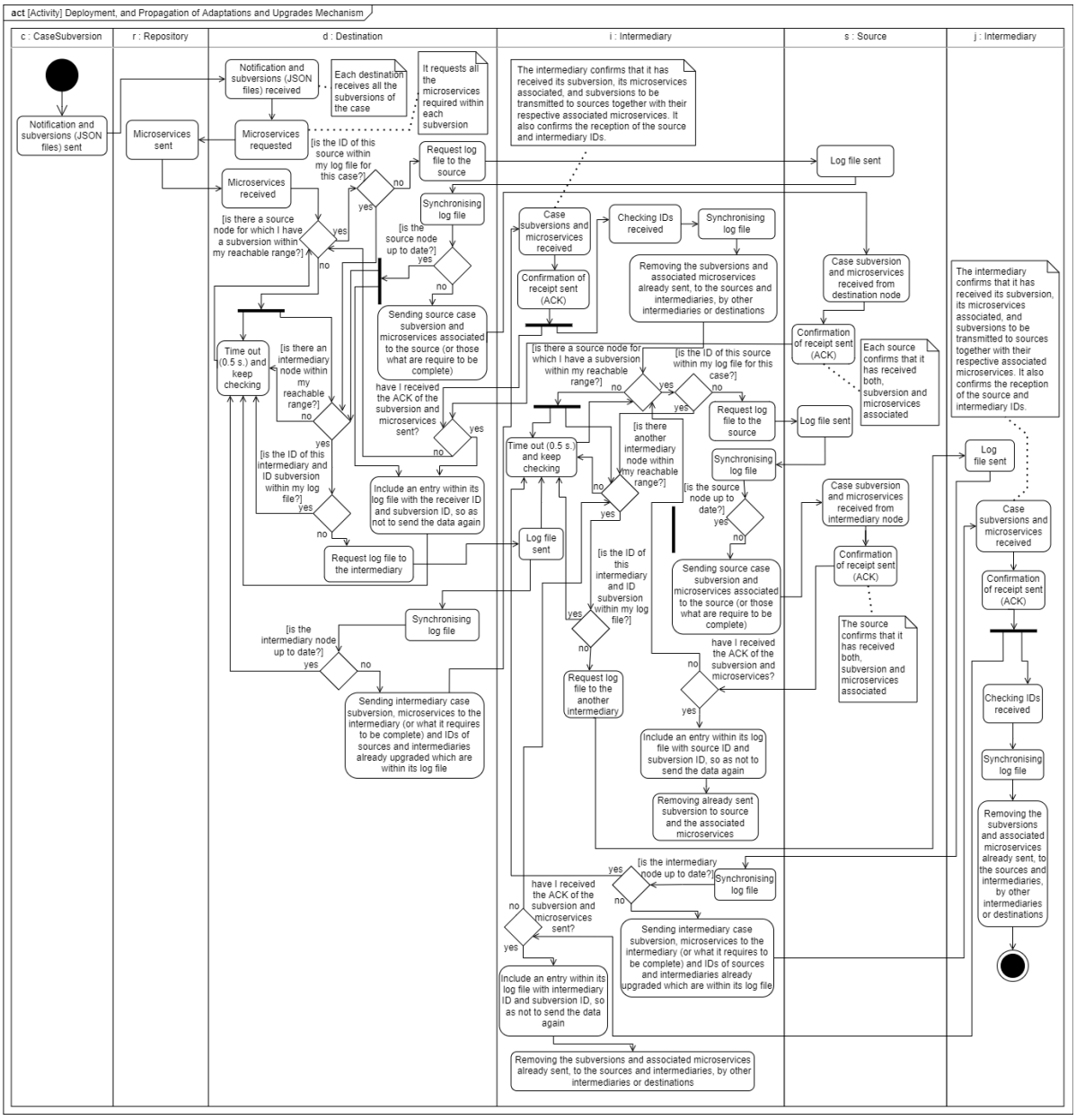


Fig. 5.9 *Activity diagram for deployment, and propagation of adaptations and upgrades mechanism in SysML.*

5.9.2 Data Gathering Mechanism

ASTREA framework supports the transmission of the data gathered from the source nodes to destination nodes through the *data gathering* mechanism. In this mechanism, ASTREA is concerning about the WSN structure and it addresses two goals (1) maximising the success rate in data gathering mechanism, and (2) maximising source nodes lifetime.

Therefore, the *data gathering* mechanism provides by ASTREA take into account the following considerations:

- Source nodes storage is limited and the time at which they will have contact with the intermediary nodes is indeterminate, since the mobility of the intermediary nodes is non-deterministic, nor is the mobility of the sources, which could also be mobile.
- Source nodes cannot send data to another source nodes. They cannot act as routers in multihop connections but if there is a destination node within reach, the sources must send data to it. Otherwise, they will connect to a reachable intermediary node to transmit the gathered data to it, and for the intermediary to send them to the destination node, via a multihop connection.
- If a source node has no connection with an intermediary or destination node, it will perform data buffering, temporarily storing the gathered data to transmit it once reconnected.
- It is assumed that intermediary nodes will eventually reach a destination node for an indeterminate time.
- If the remaining storage of the nodes is reaching a threshold, and in the case specification there is defined a prioritisation of data to be kept, their timestamp will be checked to remove those with the same priority but older.
- If there is no destination node in range (via multihop) but there are more than two intermediary nodes, an election will have to be carried out to determine which of the intermediary nodes is the most suitable to receive the data from the sources. ASTREA harnesses the use the election algorithm presented in [96] and the parameterised evaluation function defined below will be used which considers direct connections to other nodes (dc) tracked from the routing table, remaining battery (rb), remaining storage (rs) capacity, and number (n) of direct connections to other devices obtained from the routing table:

$$Ev(dc, rb, rs) = \begin{cases} 0 & \text{rs=0} \\ 0.25 \times \frac{dc}{n} + 0.25 \times rb + 0.5 \times rs & \text{rs>0} \end{cases} \quad (5.1)$$

However, different election algorithms could be used within ASTREA framework.

- When source and intermediary nodes transmit the data that they temporarily store and receivers (i.e., intermediaries or destinations) receive the data, the latter sends back an acknowledgment message (ACK). The senders remove the data sent (i.e., to free up space) from their temporary storage, once they have received the corresponding acknowledgement (ACK).
- Preprocessing actions are performed as early as possible to reduce the volume of non-useful information to be transferred over the WSN, i.e. prioritising within the nodes where the data is generated and, then on the intermediary nodes or finally on the destinations. If the intermediary nodes do not have the computational capacity to do so, the operations would finally be performed at the destination nodes.
- Several datasets can be preprocessed together coming from different sources or intermediary nodes.

The entities that participate when executing this mechanism, how they interact with each other and the execution flow, is detailed in Figure 5.10.

5.10 ASTREACE Tool

A case can be represented as a set of actions in which the data follows a flow. For example, *monitoring a person's heart rate with a reporting frequency of one minute and get the maximum of each hour*, can be represented by a flow composed of three actions: (1) collecting the heart rate measurement from the sensor each minute; (2) preprocessing these data, getting the maximum of each hour; and (3) storing the results.

In this context, the ASTREACE visual editing tool incorporates a palette with all possible elements (i.e., general nodes, environmental sensors, body sensors or wearables, preprocessing actions, storage actions, and device state) that can be utilised and from which the specification of a case can be designed. The elements can be selected, dragged and dropped onto a canvas by the case designer to build complex monitoring systems in an intuitive and high-level way, avoiding the need of low-level programming.

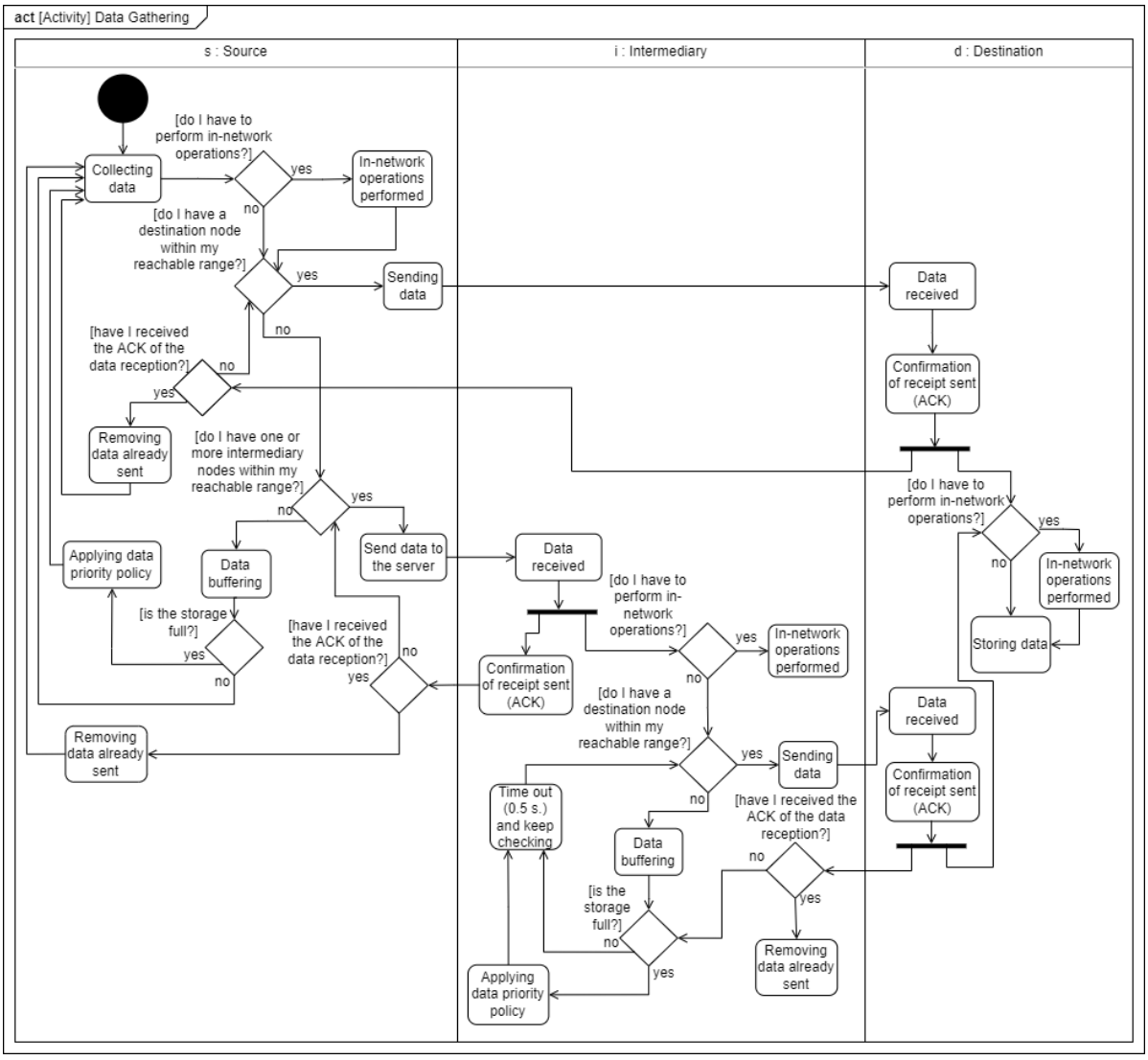


Fig. 5.10 *Activity diagram for data gathering mechanism in SysML.*

Once the case designer completes the specification, a JSON file is generated, which is what the *case subversion* (Section 5.8) works with.

All the elements included within the case specification (which are available through ASTREACE) have been previously implemented as microservices by a software developer and they will be available through the *service repository* (Section 5.11).

This not only facilitates the design of monitoring systems, but strengthens the reuse of already implemented components. Moreover, ASTREA's design allows for new elements to be incorporated in the future.

5.10.1 ASTREACE Elements

Some of the main ASTREACE elements can be found in the following subsections.

General Nodes

General nodes (Figure 5.11) includes two types of nodes:

- *Value* represents a string or number and optionally, its unit (i.e., 60 bpm meaning beats per minute) that will be the input for the next connected element. The *value* element has no input, it does have an output.
- *Print* allows a measurement to be labelled with a custom string (e.g., asleep). The *print* element has an input and an output.

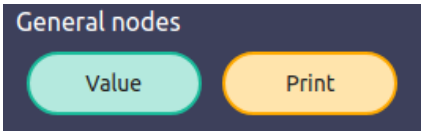
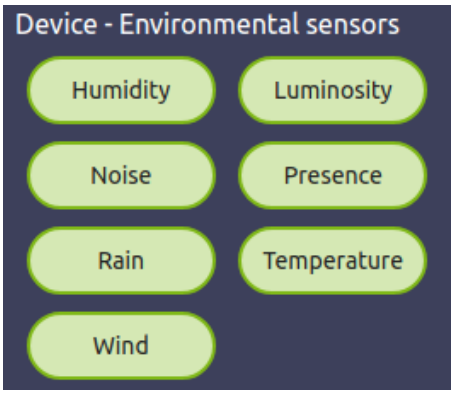


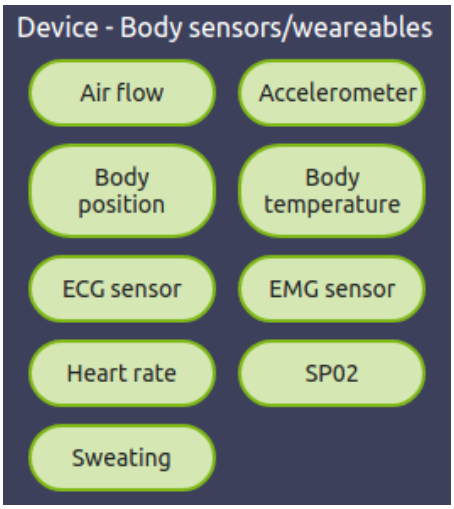
Fig. 5.11 *General nodes of the element palette of the ASTREACE visual editor.*

Device - Sensors and Wearables

ASTREA provides the possibility of incorporating *environmental sensors* and *body sensors/wearables*. *Environmental sensors* includes all the hardware devices grouped by measurement that can be used within a case. ASTREA, currently, allows the use of *sensors* (Figure 5.12a) to monitor humidity, luminosity, noise, presence, rain, atmospheric pressure, temperature and wind, and *sensors or wearables* (Figure 5.12b) to monitor air flow, acceleration,



(a) *Device (environmental sensors) of the element palette of the ASTREACE visual editor.*



(b) *Device (body sensors/wearables) of the element palette of the ASTREACE visual editor.*

Fig. 5.12 *Sensors and wearables to monitor environmental and physiological variables.*

body position, body temperature, electrical activity of the heart, electrical activity produced by muscles, blood oxygen saturation, and sweating.

Each hardware device generates at least one output but it has no input. The case designer must select within each element the identifier of the specific hardware device to be used in the monitoring systems. He/she also can specify a value to set the *reporting frequency (RF)*, although it may possible that the hardware device does not support to set a custom *RF* in a native way, this can be achieved by an implicit *preprocessing action*. When the device allows the RF to be set, this does not have to be done manually, but there are devices that allow the reporting frequency to be set by a function built into their API. If the device does not allow RF setting, it will be done by means of the *preprocessing actions* included within ASTREA.

In addition, the case designer should set a priority within the range [1-5] for each measurement, with 1 being the highest priority and 5 the lowest. This priority value represents the prevalence of the data relative to others when the temporal storage of a source or intermediary node becomes full.

Preprocessing actions

Preprocessing actions (Figure 5.13) group nine types of elements:

- *Average* refers to the arithmetic mean of a set of data within a period T.
- *Maximum (Max.)* refers to the maximum value of a set of data within a period T.

- *Minimum (Min.)* refers to the minimum value of a set of data within a period T.

  These three elements (*average*, *maximum*, and *minimum*) have an input through which they receive a data stream.

- *Filter* generates the same output value as the input value for each input value that satisfies the condition specified by the case designer in the drop-down list that includes the element.

- *Unchanged for* is a timer that returns *true* when a value has not changed by a defined time. The *unchanged for* element has an input and an output.

- *Arithmetic operator* returns a value as a result of two input values after operating with an operator (+ , -, $*$, and $\div$).

- *Relational operator* returns *true* if the condition (==, !=, <, >, $\leq$ and $\geq$) applied to two input values is satisfied.

- *Logical operator* returns *true* if the operator (AND, OR, and NOT) applied to two input expressions is satisfied.

  The case designer can select one of the operators available in ASTREACE (i.e., *arithmetic operator*, *relational operator* or *logical operator*) which work with two inputs and generate one output.

- *Count* returns the number of times that one input value or a set of them within a time period, compared to another value satisfies the condition (==, !=, <, >, $\leq$ and $\geq$).

The conditions, comparison values, and periods must be set by the case designer.

Storage actions

Storage action (Figure 5.14) includes storage element to represent the input values that will be stored in a database. It has an input connector but no output connector.

Device

Device (Figure 5.15) includes:

- *get state* that returns sensor or wearable remaining battery and available storage capacity; and

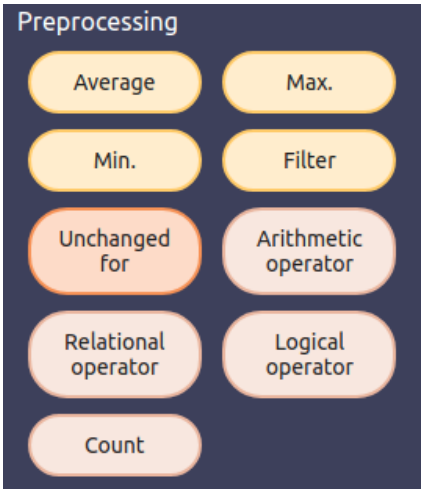


Fig. 5.13 *Preprocessing actions of the element palette of the ASTREACE visual editor.*

Fig. 5.14 *Storage action of the element palette of the ASTREACE visual editor.*

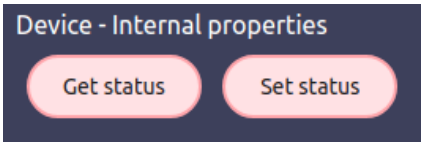


Fig. 5.15 *Device (internal properties) of the element palette of the ASTREACE visual editor.*

- *set state* that specifies the state to which the device has to change to.

Finally, ASTREACE includes the options to export an already designed case as well as to import a previously designed case. The case design is stored in JSON format.

5.10.2 ASTREACE Constraints

There is no limitation on the number of elements that compose a case and ASTREACE includes the option to set multiple configuration values within the different elements to ensure flexibility in case design. However, ASTREACE also integrates a handler to check if the connections between elements and the generated flows are valid, thus minimising errors in the design of the case and ensuring that the design of the monitoring system is feasible. The constraints included are the following:

- It is not allowed to connect an input element to a sensor or wearable.
- It is not allowed to use more than one device with the same identifier in the same case.
- It does not allow an *arithmetic operator* to be applied to a *print* element.
- It does not allow to perform an *average*, *max.* or *min.* operation on a *string*.
- It does not allow to apply some *relational operator* (i.e., <, >, $\leq$ or $\geq$) on a *string*.
- It does not allow to connect something other than an expression to a *logical operator*.
- It does not allow to connect something that does not return a `boolean` to a *set state* element.

5.11 Service Repository

ASTREA provides a repository of microservices designed to cover the requirements of the monitoring systems (cases) that can be built with ASTREA. From the case specification, the functionality is associated with the microservices to be deployed, also autonomously, on the hardware devices (nodes).

Developed microservices encapsulate the functionality of each particular ASTREA action, these are hosted within the *service repository* to be reused with different measurements. Some of them are also available for multiple platforms. It should be noted that ASTREAMS framework can be extended with new elements and configuration options derived from new requirements or application scenarios.

5.12 Case Study

Here, we aim to animate the capabilities of the ASTREA framework and the interaction between the primary components from an example based on the design of a case study, its deployment in a mobile and dynamic network and its subsequent adaptations and upgrades. It also describes the data gathering mechanism. In relation to modelling, the case study is related to the Use Case View within the 4 + 1 View Model.

5.12.1 Case Design

First of all, it is necessary to think about the requirements of the monitoring system. The designed system is intended to cover the following requirements:

- **FR.1** monitor environmental temperature every 10 minutes;
- **FR.2** identify where (e.g., in which room) it is the hottest every hour (average and maximum per hour);
- **FR.3** monitor noise and luminosity continuously;
- **FR.4** count how many times a noise of 40 dB (equivalent to the noise of a conversation) per hour is exceeded;
- **FR.5** detect the average luminosity of the rooms;
- **FR.6** monitor body temperature every 10 minutes;
- **FR.7** identify the maximum and minimum body temperature per hour;
- **FR.8** monitor heart rate to measure beats per minute (bpm);
- **FR.9** count how many times the rate of beats per minute (bpm) is equal to or greater than 120 bpm in an hour;
- **FR.10** monitor air flow to measure the number of breaths per minute; and
- **FR.11** detect the average hourly air flow.

Second, the case designer has to select each of the sources (sensors or wearables) that he or she wishes to incorporate and the actions that the system should satisfy based on the system

requirements. In this case study, we will use ASTREACE visual editing tool for the design, although it is not mandatory.

Sensors or wearables are required to collect the physiological and environmental data. Therefore, the case designer must select them from the palette, drag and drop them on the canvas, which of the elements available in ASTREA he/she wishes to include in the system. The case specification that a case designer has initially designed is shown in Figure 5.16. The device elements (green boxes) are the selected measurements. Each measurement has a set of associated devices available within ASTREA that allow for the collection. The case designer must select from the drop-down menu the sensor identifier he/she wishes to use. The reporting frequency (RF) of each sensor can be configured at software level. For this case, it has been set to 10 minutes for the temperature sensors and to 0 minutes for the rest, following the requirements of the case. Zero corresponds to the default data collection of the sensors (continuous monitoring). On the basis of these adjustments, the requirements RF.1, RF.3, RF.6, RF.8, and RF.10 would already be satisfied. Nevertheless, the case designer also sets the data priority because he/she considers that ambient temperature is more relevant than noise and luminosity, and likewise, heart rate and air flow is more relevant than body temperature.

The selected devices by the case designer and a possible visual allocation within a scenario can be seen in the Figure 5.17). Sources A, C, D and E include temperature, luminosity, and noise environmental sensors. Moreover, source A includes a presence sensor. The case designer has selected temperature, luminosity, and noise sensors, devices with the same computational capabilities for pair D and E but different from those for pair A and C. Source B will monitor a patient's body temperature, heart rate and air flow. The case designer does not have to worry about which intermediaries or destination nodes will be included in the scenario, or how many of them will be involved in the monitoring system, neither include them in the case specification.

The case specification includes preprocessing elements to get independently the maximum, minimum, and average (yellow boxes) of each measurement they receive as input. The dependent elements (sources and actions) should be connected by arrows. Within these elements, the design must establish the period or time window for selecting the data on which the operation is carried out. In this case, the interval is 1 hour. These adjustments would satisfy the requirements RF.5, RF.7, and RF.11. It also should be noticed that there is one maximum action and one store action (i.e., *global actions*) which have as inputs the outputs of other four actions of maximum in order to fulfill the requirement RF.2.

Within the preprocessing elements also the counting operations are included. The counting operations, for this case, have as inputs the noise device and another one the heart rate to satisfy

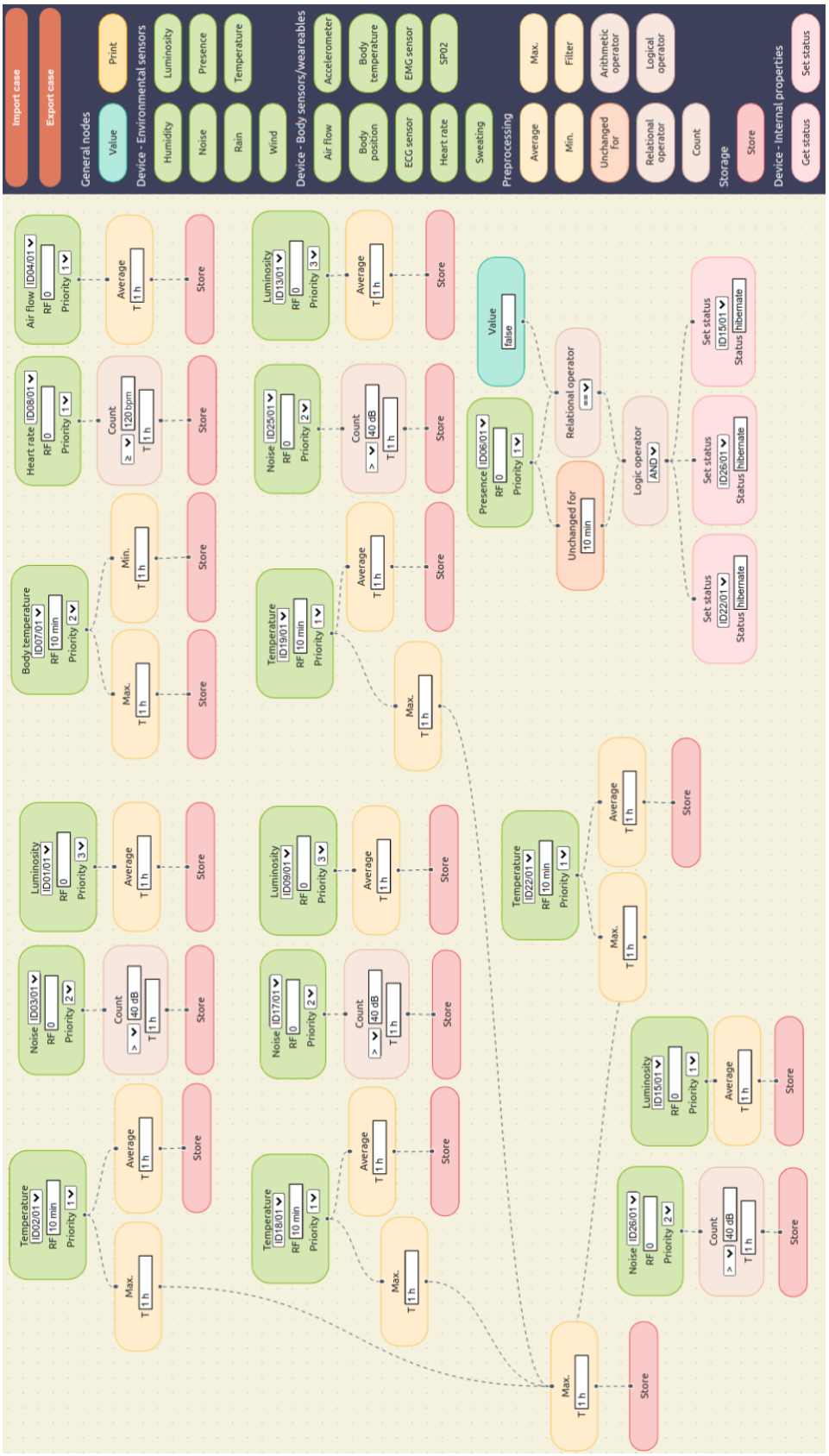


Fig. 5.16 *Case specification with ASTREACE tool.*

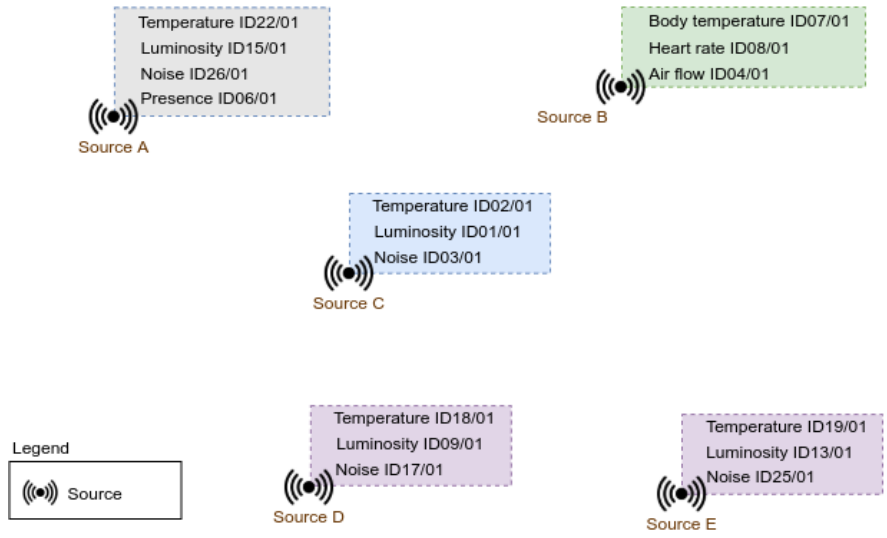


Fig. 5.17 *Sources selected with ASTREACE and visual allocation within a scenario.*

the FR.4 and FR.9 requirements respectively. Storage actions have been added to save the data after it has been processed, and consequently to be transmitted to the destination nodes.

This part corresponds to the functionality of the system related to monitoring. Changes, after deployment can be incorporated through ASTREACE, and is what is considered an adaptation or an upgrade. With automated deployment, these changes can be framed as adaptive system changes.

Up to this point, the functionality of the system would be defined, which focuses on monitoring the state of the users and their environment, in order to find out how it might affect them. However, ASTREA includes the option to monitor the environment and the state of the system to improve its performance in terms of bandwidth and energy saving. For the latter, the designer of the case, in one of the rooms, has included a presence device in order to hibernate noise (ID26/01), luminosity (ID15/01) and temperature (ID22/01) devices when no one is in the room for 10 minutes. The presence device has a reporting frequency set to 0 (i.e., default reporting frequency of this device). Its output is combined with the *value* element set to `false` using the *relational* == *operator* to check that there is no one in the room (which is empty). Presence device output is the input of the *unchanged for* action from which it is intended to detect that the value collected by the sensor does not change for 10 minutes. The two operators are combined through the *logical AND operator* to finally hibernate the devices. Such changes, which the system performs autonomously, can be framed as self-adaptive system changes.

Once the case is designed with ASTREACE visual editor, the specification is managed by the *case subversion* which generates a particular JSON file for each source, another JSON file that would be deployed on intermediary nodes if any, and a JSON file to be deployed in

destination of which there has to be at least one to start the *deployment* mechanism. All those files are managed as subversions of the case specification generated by ASTREACE visual editing.

In this scenario, the sources D and E have the computational capabilities to perform *max.*, *average*, *count* and persistent *store* actions but sources A and C do not. Therefore, the *case subversion* service generates the following subversions (JSON specification files) for the sources accordingly which are illustrated in Figure 5.18.

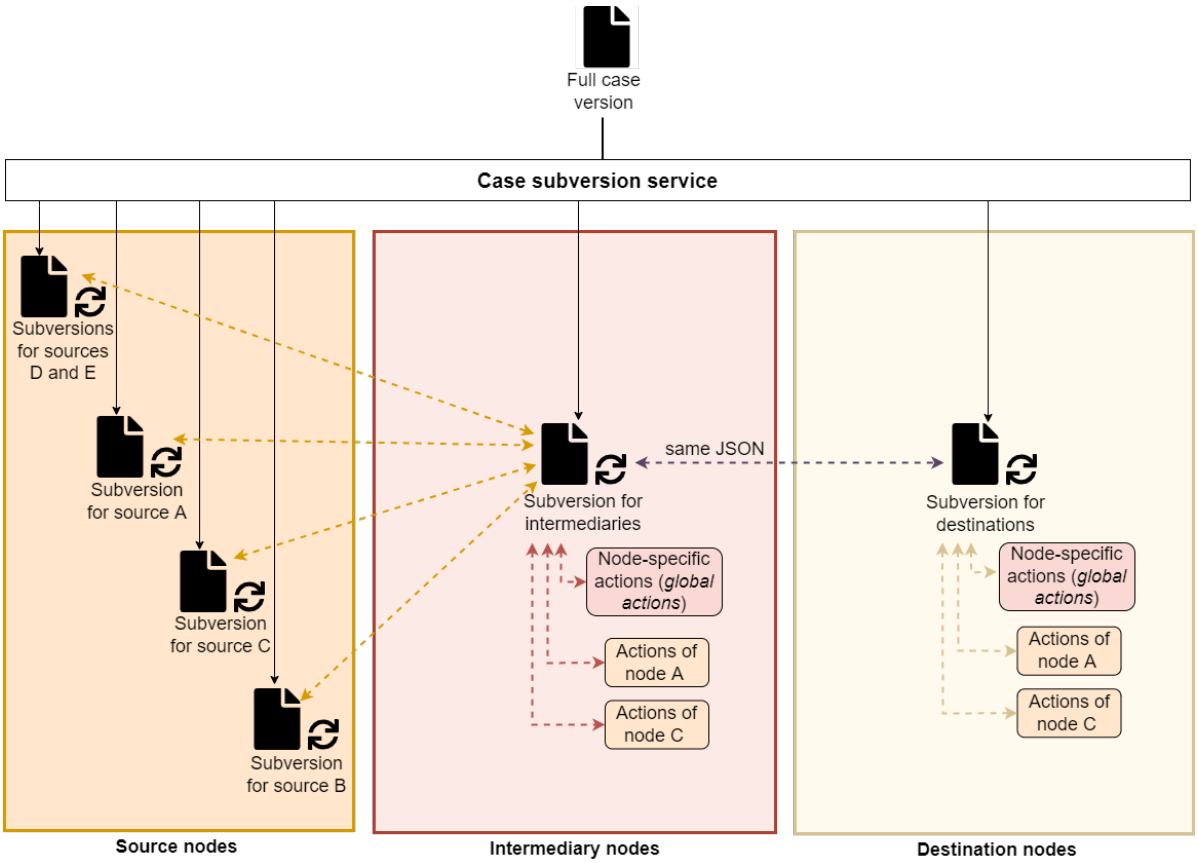


Fig. 5.18 *Node subversions generated by case subversion service from a case version specification.*

- An equal *subversion for sources D and E* because source nodes D and E are equipped with the same sensors, and therefore, the same case specification is valid for both sources. These two source nodes have sufficient capabilities to perform in-network preprocessing.
- A *subversion for source A* which is different from the one *subversion for source C* because the nodes are equipped with different sensors. These two source nodes do not

have sufficient capabilities to perform in-network preprocessing but they can send raw data.

- A *subversion for source B* which has capabilities to perform in-network preprocessing.

In this particular scenario, intermediary nodes have sufficient capabilities to perform in-network preprocessing, and the subversion for intermediary and destination nodes are shared. Actions should be carried out as soon as possible, in order to minimise the volume of data transmitted. *Max.*, *average* and *count* actions are not supported at sources A and C, and therefore these source nodes will gather the data to the intermediary nodes where they will be performed. With regard to the two global actions (i.e., *max.* and *store*) to be carried out on the basis of environmental measurements, they will be carried out as a first option on the basis of the intermediaries and as a second option at the destination nodes. The following situations may occur:

- If an intermediary collects data from two or more sources but not from all four environmental sources of the scenario, it is already taking actions on the basis of the data it has available. The resulting volume of data will be smaller than the volume of raw data generated, and will be transmitted to the destination nodes.
- The same actions may be taking place in other intermediary nodes with other data.
- Results of actions (i.e., *max.* and *store*) performed within different intermediaries will be gather to the destination nodes where the global action will take place again for the whole set of data. This occurs because intermediaries do not have all the data required to complete the actions.

5.12.2 Deployment Case Subversions

Here, we aim to illustrate the *deployment, and propagation of adaptations and upgrades* mechanism supported by ASTREA framework. Figure 5.19 shows the position of the source devices, an example of how the interaction with two intermediaries and a destination could look like.

Firstly, *destination A* receives a notification to star the deployment and the subversions of the case. Secondly, the *destination A* requests the download of the microservices that are included within each subversion to the *service repository*. These microservices will perform the monitoring actions, preprocessing, storage, and management of the device defined within each subversion.

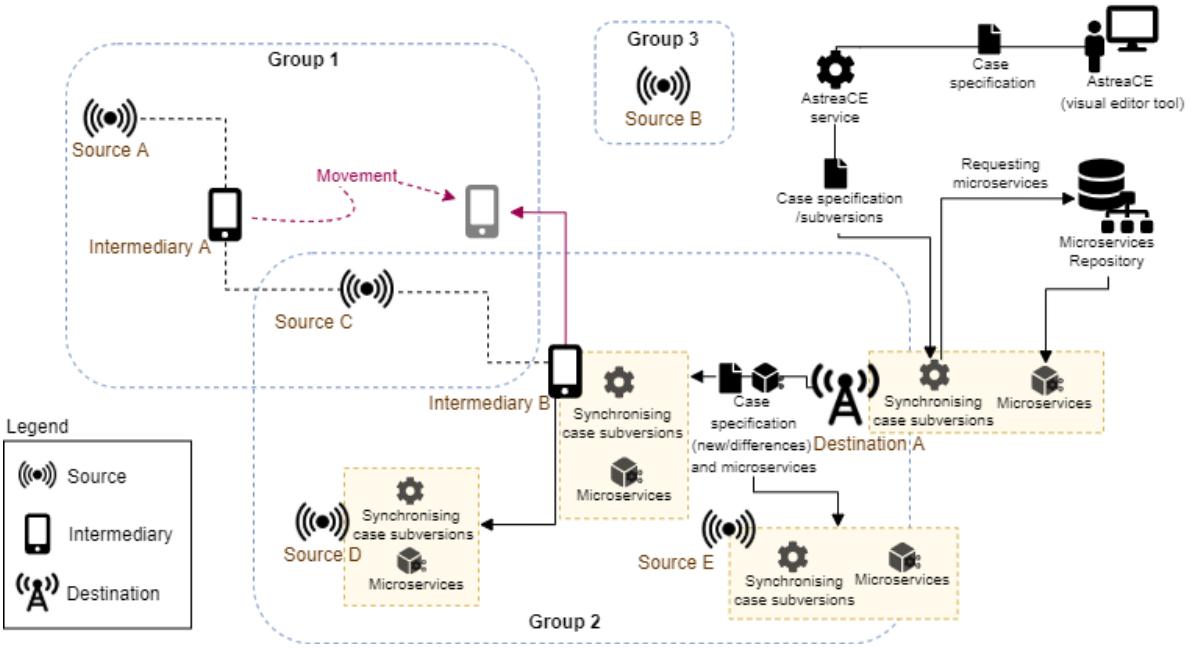


Fig. 5.19 *Deployment, and propagation of adaptations and upgrades mechanism.*

Destination A acts as the gateway of the WSN and initiates the transmission of subversions and microservices to the other nodes. As a third step, *destination A* checks which source nodes are within its reach. Here, only the *source E* is within its reach and therefore *destination A* checks if it has the source *ID* along with the subversion that needs to be deployed within its log file which means that the adaptation or the upgrade has already been performed. It is the initial deployment for this case, and therefore, the *destination A* does not have the *source E ID* within its log. However, *destination A* requests *source E* log file to check what has the source node deployed because the deployment could have been started previously but the connection may have failed or been interrupted, and the update process may not have been completed.

After *destination A* synchronise the log files, it sends the specification (case subversion) to the *source E*. *Source E* sends back an ACK to confirm receipt of the data (i.e., subversion and microservices), and *destination A* does not retransmit this data to other intermediary nodes.

Destination A also has to send the specification (subversions) and microservices associated to *sources A*, *B*, *C*, and *D* but such nodes are not within its reach. Nevertheless, there is an intermediary on the move in the scenario (*intermediary B*) that has entered within the range of the *destination A*, and at that time, *destination A* establishes a communication to send it the subversions and microservices of *sources A*, *B*, *C* and *D*. Before sending, the *destination A* checks that *intermediary B*, is not up to date. To do this, the *destination A* checks that there is no an entry in its log file for the *ID* of the *intermediary B* and the subversions it has to send to

it. As there is none, it also requests for its log file, because the *intermediary B* might already has some of the information (subversions and microservices) from a previous adaptation or upgrade attempt. *Destination A* sends subversions, associated microservices and *source E ID* (because the latter is already upgraded).

Intermediary B sends back an ACK to confirm receipt of the data and *destination A* updates its log file with an entry of the information sent in order not to send the same information repeatedly. Whether *destination A* does not receive ACK for all transmitted data, the next time *intermediary B* reaches it, *destination A* will resend the corresponding data to it.

Intermediary B continues its round of work by getting *source D* into its range. Therefore, *intermediary B* checks if the *source D ID* is within its log because the data transmission might have been initiated previously by itself but not yet completed because a disconnection or because this intermediary does not know if there are other intermediaries in the scenario that might have already upgraded *source D* case information. After checking it, *intermediary B* sends the subversion that corresponds to *source D* and associated microservices. *Intermediary B* expects to receive an ACK to confirm receipt of the data. In this scenario, *source D* sends the confirmation ACK and *intermediary B* updates its log file with an entry of the information sent to *source D*. Subsequently, *intermediary B* removes the specification corresponding to *source D* and the associated microservices. Whether *intermediary B* did not receive the ACK, e.g., because of a disconnection, *intermediary B* would not remove the subversion and microservices for *source D* but the *intermediary B* would redo the checks again, staring with whether the *source D* is still in its scope range.

The mobility of intermediaries is not deterministic but opportunistic. The *intermediary B* continues to move and the *source C* enters its range. The deployment mechanism outlined for *source D* is repeated for this *source C*. There is another intermediary (*A*) within the scenario but *Intermediary B* does not have it within its reachable range, but as soon as it does, it will check if it has the *ID* of this *intermediary A* within its log along with the *ID* subversions for this case. If it did, the *intermediary A* already has its corresponding subversions and associated microservices, then its case information is already upgraded. However, once the are reachable, both check but *intermediary B* is the one with the case information. This time *intermediary B* does not have an entry within its log file confirming that the *intermediary A* is up to date. *Intermediary B* requests the log file to the *intermediary A* to check what the node currently has and synchronise the logs (i.e., if *intermediary A* had to send the information to *source C* it is no longer necessary because it has already been upgraded). In this scenario, *intermediary A* has not yet interacted with any other node. Therefore, *intermediary B* sends to *intermediary A* all

case subversions, associated microservices, and node IDs (i.e., *IDs* of sources C and *D*) already upgraded.

Source A will be upgraded by *intermediary A* when it comes within range of the *intermediary A*. Similarly, *source B* will be upgraded, but to do so, it must come within range of one of the intermediaries. If the intermediary nodes were to meet again, they would synchronise their log files and delete the information already delivered to the source nodes with which they have already interacted (e.g., *source A*, *C*, and *D*) at some previous point in time.

5.12.3 In-network Preprocessing and Data Gathering

Here, we aim to illustrate the *data gathering* mechanism supported by ASTREA framework. Figure 5.20 shows the position of the source devices, an example of how the interaction with two intermediaries and a destination could look like.

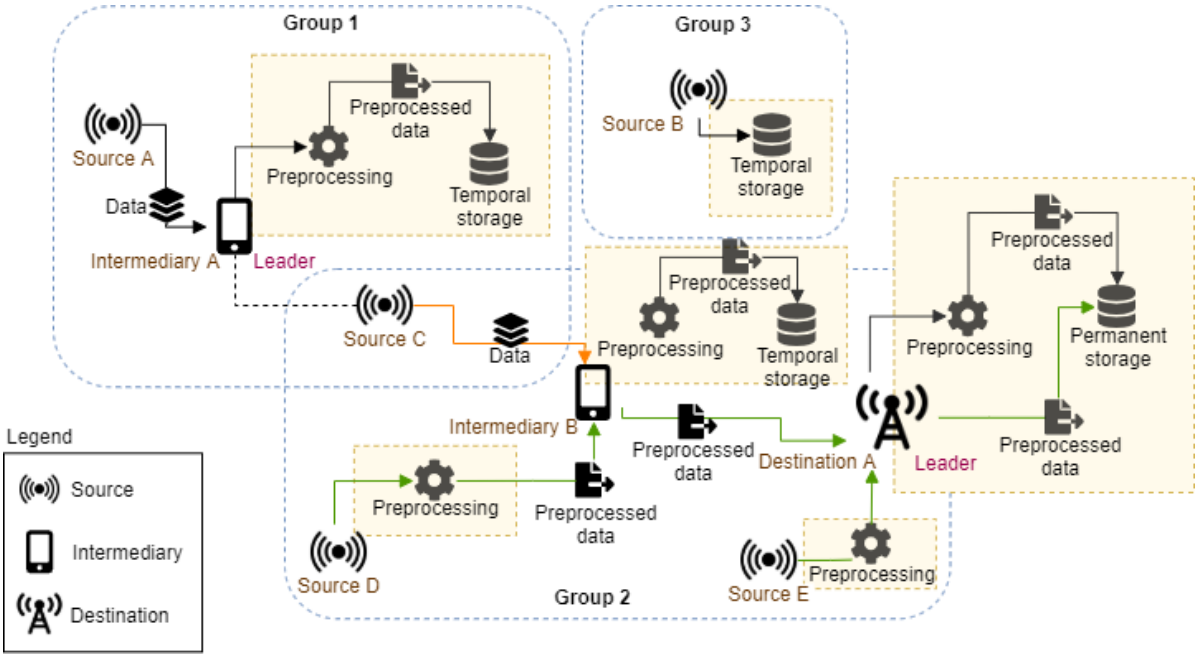


Fig. 5.20 *Data gathering mechanism.*

The actions defined within the subversion generated for each node are performed. The *sources A* and *C* do not have sufficient computational capacity, and therefore data are transmitted in raw form to intermediaries (or destinations) whether they are in range. To perform these actions that cannot be performed at the source nodes (i.e., *sources A* and *C*) themselves, the actions are added to the case subversions generated for intermediaries (if any) and destinations, so that the actions could be performed as quickly as possible and the volume of data to be

transmitted could also be reduced as soon as possible in the transmission flow. In this scenario, *source A* transmits the raw data to *intermediary A* and it is the latter that carries out the actions (i.e., *max*, *average*, *count* and *storage*) defined by the case designer for the measurements of the sensors*ID22/01 (temperature)*, *ID26/01 (noise)*, *ID15/01 (luminosity)* and *ID06/01 (presence)*. Subsequently, when *intermediary A* will be within range of *destination A*, *intermediary A* will transmit to it only the useful information corresponding to the data already processed. The same will be applied for sensor data collected in the *source C* although such data will be sent to *intermediary B*, because the *data forwarding policy* so establishes it since *intermediary B* is more suitable at this time than *intermediary A*.

In this scenario, *sources D* and *E* do have sufficient computational capacity to perform the actions defined by the case designer. Therefore, the *case subversion service* maintains their specifications (JSON file subversions) performing the preprocessing actions within the nodes themselves prior to sending the data to the intermediary or destination nodes.

Intermediary B therefore receives the raw data from *source C* and the already preprocessed data from *source D*. *Intermediary B* has within its subversion, the specification of the actions that must apply on the data from *source C* to complete these that cannot be done by itself. The *intermediary B* also includes its own specification as an intermediary node, so subsequently it must perform the actions that correspond to it (i.e., global actions), in this case, calculating the global maximum temperature, as the preprocessing in the network must be done as soon as possible. The preprocessed data by *intermediary B* are sent to *destination A* who must perform the actions defined within its specification. *Source B* has no intermediary and no destination within its range, so it must store the data it generates in its temporary storage.

Source E has in its range both *intermediary B* and *destination A*, in this case, the *data forwarding policy* states that data are sent directly to *destination A*. In this scenario the *destination A* must calculate the global maximum (of the four temperature sensors) in overlapping time windows (same 1 hour period).

Each time a sender sends data, it expects to receive from the receiver an ACK message confirming receipt. Once the ACK is received, the sender removes the data to free up space. It may occur that the temporary storage of one of the nodes (sources or intermediaries) becomes full, in this situation, the data with lower priority will be removed (i.e., first the data with priority 3 and then those with priority 2), and whether the priority matches, the oldest data will be removed.

5.13 Implementation and Feasibility of Deployment

An implementation and deployment in a test environment has been carried out to animate the capabilities of ASTREA framework. This implementation corresponds to the Implementation View within 4+1 View Model while the deployment corresponds to the Deployment View.

Firstly, a *case* has been designed with ASTREACE visual editing tool which will be deployed within the network infrastructure. In Figure 5.21 can be seen a deployment diagram of the five devices (nodes) that are part of this network infrastructure, prior to the deployment of the *case*. Two of these nodes are part of the Internet or could be deployed in a cloud infrastructure, thus are out of the WSN. In particular, they are two independent servers containing the *service repository* and the *case subversion service*. The deployment has been done in this way to demonstrate the decoupling of the services and that the operation is correct, similarly both could be on the same server. The servers run the Ubuntu Server 20.04 operating system.

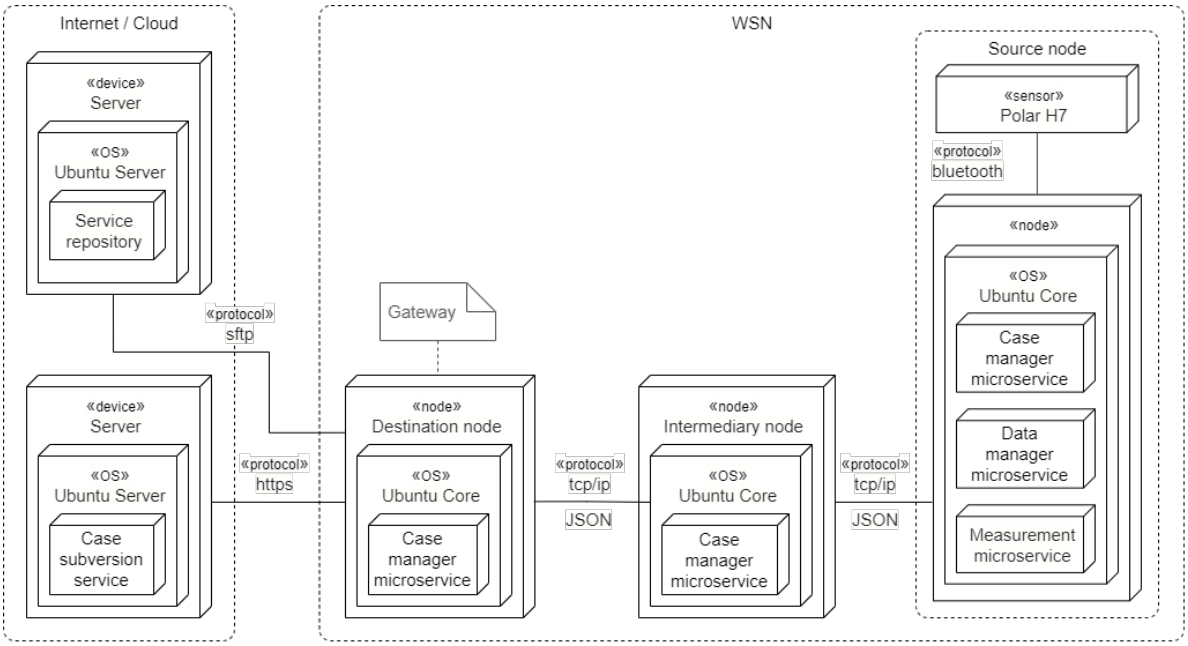


Fig. 5.21 *Deployment diagram of the devices that are part of the network infrastructure, before deploying the case.*

Additionally, there are three nodes that compose the WSN. Each of them has been assigned one of the roles defined in ASTREA. There is a *destination* node, which acts as a Gateway between the WSN and the Internet/Cloud; an *intermediary* node and a *source* node. The *source* node is connected via Bluetooth to a Polar H7 heart rate sensor [188]. The WSN nodes run the

Ubuntu Core 20 operating system[1], a lightweight operating system designed for IoT devices. The WSN nodes communicate with each other via TCP/IP connections. The *destination* node communicates with external services (hosted on the servers) via HTTPS and SFTP for downloading the services needed to compose the *case*. For this particular implementation and deployment, the services and components included within ASTREA framework have been implemented in Python 3.8.

Initially, all network nodes have deployed the *case manager microservice*, which acts as orchestrator and will make possible the autonomous deployment of the designed *case*. According to framework terminology, the *case manager microservice* can be considered a frozen spot.

Additionally, the *source* node has a *measurement microservice*, which makes it possible the communication via Bluetooth with the Polar H7 sensor. However, the node associated to Polar H7 sensor, which together make up the *source* node, does not have enough computational capabilities to carry out preprocessing operations, so these must be performed in the *intermediary* node. Therefore, *source* node will send the raw data to the *intermediary* node.

The most relevant parts of the microservice code can be found in the Code 5.1. In this code, it can be seen how using the Python Gatt library[2], a connection is established with the Bluetooth device (i.e., Polar H7 heart rate sensor) by means of its MAC address, stored in the configuration of the node. In addition, it can be seen how heart rate readings are received from the sensor.

Code 5.1 Measurement microservice for Polar H7 heart rate sensor in Python

```
manager = gatt.DeviceManager(adapter_name='hci0')

class AnyDevice(gatt.Device):
    _UUID_SERVICE_DEV_INFO = '0000180a-0000-1000-8000-00805f9b34fb'
    _UUID_SERVICE_BATT = '0000180f-0000-1000-8000-00805f9b34fb'
    _UUID_SERVICE_HR = '0000180d-0000-1000-8000-00805f9b34fb'

    _UUID_CHARACTER_FIRMWARE_VER = '00002a26-0000-1000-8000-00805f9b34fb'
    _UUID_CHARACTER_BAT_LVL = '00002a19-0000-1000-8000-00805f9b34fb'
    _UUID_CHARACTER_HR_MEASURE = '00002a37-0000-1000-8000-00805f9b34fb'

    def services_resolved(self):
        super().services_resolved()
```

[1] https://ubuntu.com/core
[2] https://github.com/getsenic/gatt-python

```
            for s in self.services:
                if s.uuid == self._UUID_SERVICE_HR:
                    for c in s.characteristics:
                        if c.uuid == self._UUID_CHARACTER_HR_MEASURE:
                            c.enable_notifications()

    def characteristic_value_updated(self, characteristic, value):
        if characteristic.uuid == self._UUID_CHARACTER_HR_MEASURE:
            data_manager.register(value[1])

device = AnyDevice(configuration.sensor_mac, manager=manager)
device.connect()

manager.run()
```

The *case* to be deployed is shown in Figure 5.22 and Code 5.2. It describes how the *source* node should record data from the heart rate sensor at a frequency of 1 minute. Subsequently, the *preprocessing microservices* will perform the average and maximum operation on these data in 15-minute time windows. It should be noted that the *case* specification shown in Code 5.2 includes both aesthetic and functional information. That is, the `"position"` key has the sole purpose of positioning the element on the canvas of the visual editing tool (ASTREACE) to show a human-readable version and maintain the visual aspect of the composition of the *case*. This information is not transmitted to the nodes of the network in the *case* deployment.

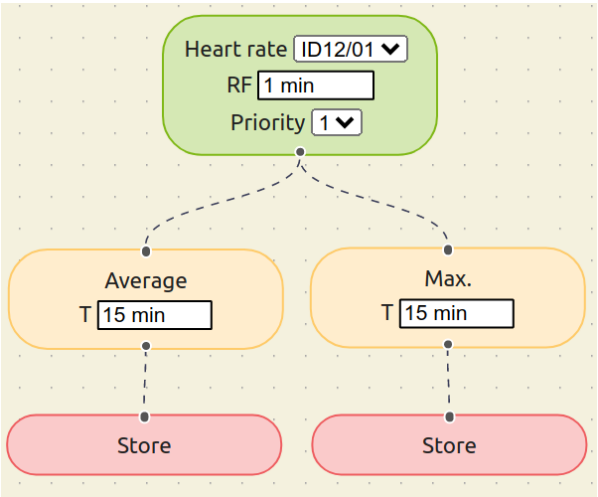


Fig. 5.22 *Case specification for proof of concept with ASTREACE visual editing tool.*

When the *case* is introduced in the system, the *case subversion service* sends the different specifications (source subversion, intermediary and destination subversions) to the *destination*

node. This will download the required microservices via SFTP from the *service repository*. Subsequently, the *destination* node will transmit the subversions and microservices to the *intermediary* node because in this implementation and deployment, we assume that the *source* node is not within reachable range of the *destination* node. The services are streamed in gzip-compressed form. Likewise, *intermediary* node will transmit the particular specification (subversion) of the *case* and microservices associated to the *source* node. The final *case* deployment is shown in Figure 5.23.

Code 5.2 Case specification in JSON generated from ASTREACE visual editing tool from specification in Figure 5.22.


```
{
        "actions":[
                {
                        "id":"node_0",
                        "type":"measure",
                        "position":{"x":90,"y":46},
                        "data":{
                                "sensorName":"Heart␣rate",
                                "sensorId":"ID12/01",
                                "MAC":"00:22:D0:DA:7D:25",
                                "rf":"1␣min"
                        }
                },
                {
                        "id":"node_1",
                        "type":"preprocessing",
                        "position":{"x":993,"y":193},
                        "data":{
                                "preprocessingOperation":"Average",
                                "T":"15␣min"
                        }
                },
                {
                        "id":"node_2",
                        "type":"preprocessing",
                        "position":{"x":183,"y":192},
                        "data":{
                                "preprocessingOperation":"Max.",
                                "T":"15␣min"
                        }
                },
```

```
                {
                        "id":"node_3",
                        "type":"storage",
                        "position":{"x":992,"y":297}
                },
                {
                        "id":"node_4",
                        "type":"storage",
                        "position":{"x":184,"y":297}
                },
        ],
        "links":[
                {
                        "source":"node_0",
                        "target":"node_1"
                },
                {
                        "source":"node_0",
                        "target":"node_2"
                },
                {
                        "source":"node_1",
                        "target":"node_3"
                },
                {
                        "source":"node_2",
                        "target":"node_4"
                },
        ]
}
```


It is worth noting that in this particular scenario, the Polar H7 sensor does not allow configuring a custom reporting frequency (RF). Therefore, to comply with the *case* specification, the *data manager microservice* (`"data_manager"` of Code 5.2) will collect, format (e.g., as Code 5.4 or Code 5.6) and retransmit the data generated by the sensor at the RF specified in the *case* specification, ignoring the rest.

The orchestrator entity (i.e., *case manager microservice*) obtains the parameters to pass to the *preprocessing microservices* from the *case* specification itself. To facilitate service orchestration at runtime, *preprocessing microservices* provide a standardised interface. The interface receives two parameters, both in JSON format (`"execute(parameters, data)"` of code examples 5.5 and 5.7). The first parameter contains the parameters required by the

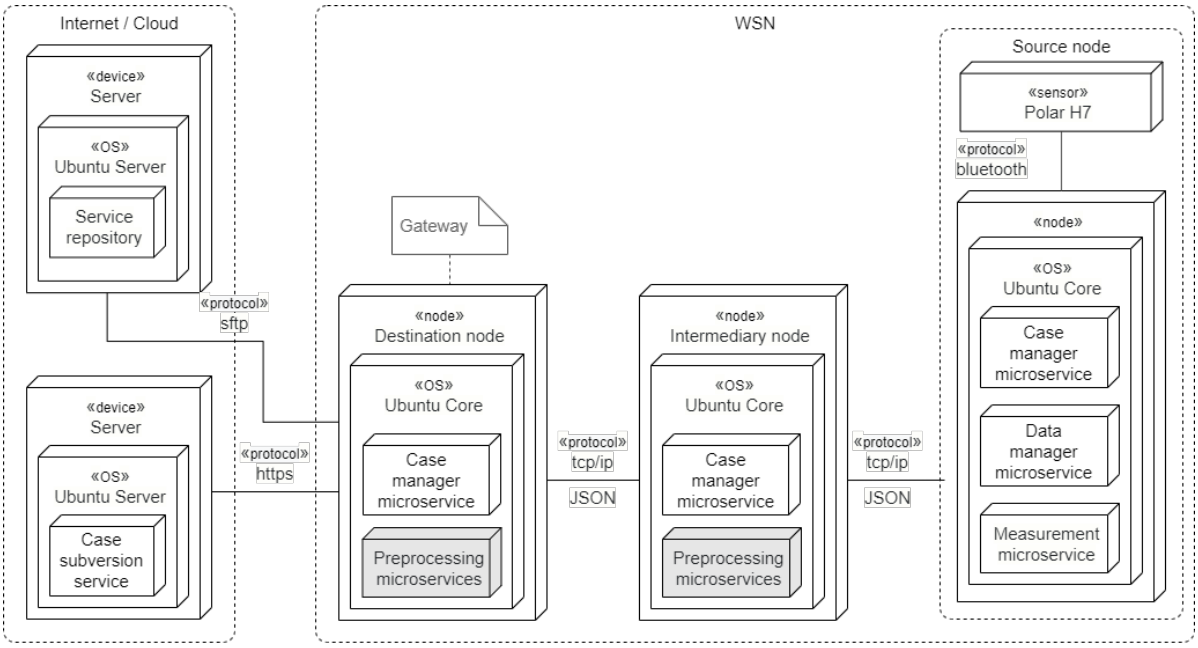


Fig. 5.23 *Deployment diagram of the devices that are part of the network infrastructure, after deploying the designed case.*

specific *preprocessing microservice*. The *preprocessing microservice* itself is responsible for checking that the required parameters are contained in the JSON passed as a parameter and that these parameters are in the required format. The second parameter contains the data to be preprocessed, as well as metadata (e.g., `"sensorId"` and `"rf"`) about these data. The *preprocessing microservices* are part of the hot spot and it allows to the developers to extend and customise ASTREA functionality following the standardised interface.

In the scenario deployed, the *average preprocessing microservice* will be invoked with the parameter described in Code 5.3. In this *case*, the microservice only requires the period (T) with which the data must be grouped in order to apply the required operation.

Code 5.3 Required parameters for the average preprocessing microservice.

```
{
    "T": "15␣min"
}
```

Regarding the data, the *average preprocessing microservice* will receive the data gathered by the *source* node. These data will be in a standardised format for ASTREA, as can be seen in Code 5.4. The JSON contains the metadata of the data and within the `data` key is the data array, where for each measurement taken a POSIX timestamp in seconds is associated.

Code 5.4 Data gathered by the source node from the Polar H7 heart rate sensor.

```
{
    "dataType": "raw",
    "sensorName": "Heart␣rate",
    "sensorId": "ID12/01",
    "rf": "1␣min",
    "data": [
      {"timestamp": 1614071934,
       "value": 74
      },
      {"timestamp": 1614071994,
       "value": 73
      },
      {"timestamp": 1614072054,
       "value": 74
      },
      [...]
    ]
}
```

Code 5.5 shows the most relevant parts of the *average preprocessing microservice*. In the first part of the code, the checking of the received configuration parameters is carried out. In this scenario, it is checked that the period parameter (T) is in the JSON, is not empty and is in the correct format.

Subsequently, by means of the `split_data` auxiliary function, the partitioning of the data into the indicated period is carried out. The function splits the data starting at 00:00 hours. If a time window is not complete, this data cannot be preprocessed. As discussed in previous sections, due to the dynamicity of the network, a set of data can be gathered by different nodes. For this reason, data belonging to the same time window may be distributed among different nodes. Performing preprocessing operations on incomplete data sets may lead to erroneous preprocessed values. For this reason, when the data set of a time window is not complete, it is not preprocessed, waiting to be completed in this node or in a higher role node in the hierarchy of data forwarding. Whereas complete data sets are returned in an array in the variable `data_bundles`, incomplete data sets are returned in the variable `remaining_data`.

Finally, when the preprocessing operation is carried out on the data, each of the resulting measurements is associated with the average of the timestamp of the grouped measurements.

Code 5.5 Average preprocessing microservice in Python

```
def execute(parameters, data):
        parameters = json.loads(parameters)
```

```
# Checking that the parameters required for the service
# are received, not empty and in the correct format.
if 'T' not in parameters:
        raise ValueError("Parameter 'T' not found.")
if parameters['T'] is empty:
        raise ValueError("Parameter 'T' empty.")
if not check_format(parameters['T']):
        raise ValueError("Parameter 'T' has an unrecognisable format.")

# Data is divided according to the period indicated.
data = json.loads(data)
data_bundles, remaining_data = split_data(data, parameters['T'])

# The metadata of the new data to be obtained is added.
procesed_data = {}
procesed_data['preprocessingOperation'] = "Average"
procesed_data['T'] = parameters['T']
if 'sensorId' in parameters:
        procesed_data['historical'] = [parameters['sensorId']]
else:
        procesed_data['historical'] = [parameters['historical']]
procesed_data['historical'].append(procesed_data['preprocessingOperation'])
procesed_data['data'] = []

# Data is processed according to the type of operation
# implemented by the microservice.
for bundle in data_bundles:
        avg_data = 0
        avg_timestamp = 0
        for datum in bundle['data']:
                avg_data = avg_data + datum['value']
                avg_timestamp = avg_timestamp + datum['timestamp']
        size = len(bundle['data'])
        processed_data = avg_data/size
        processed_timestamp = avg_timestamp/size
        procesed_data['data'] = {"value": processed_timestamp, "data": processed_data}

return procesed_data, remaining_data
```

The data returned by this microservice is shown in Code 5.6. It is important to highlight that new metadata has been added, including the history of operations performed on the data.

Code 5.6 Data preprocessed by the average microservice.

```
{
    "dataType": "processed",
    "preprocessingOperation": "Average",
    "T": "15 min",
    "historical": ["ID12/01", "Average"],
    "data": [
      {"timestamp": 1614072354,
       "value": 74.5
      },
      {"timestamp": 1614073254,
       "value": 76.1
      },
      [...]
    ]
}
```

The *maximum preprocessing microservice* performs the operation of obtaining the maximum recorded value of the data gathered in a given period. The most relevant parts of the code are shown in Code 5.7. As can be seen, like the average microservice it complies with the standardised interface to enable the orchestration of services at runtime.

Code 5.7 Max preprocessing microservice in Python

```
def execute(parameters, data):
        [...]

        # Data is divided according to the period indicated.
        data = json.loads(data)
        data_bundles, remaining_data = split_data(data, parameters['T'])

        # The metadata of the new data to be obtained is added.
        procesed_data = {}
        procesed_data['preprocessingOperation'] = "Max"
        [...]

        # Data is processed according to the type of operation
        # implemented by the microservice.
        for bundle in data_bundles:
                max_data = -sys.maxint-1
                avg_timestamp = 0
                for datum in bundle['data']:
                        if max_data < datum['value']:
```

```
                max_data = datum['value']
            avg_timestamp = avg_timestamp + datum['timestamp']
        size = len(bundle['data'])
        processed_timestamp = avg_timestamp/size
        procesed_data['data'] = {"value": processed_timestamp, "data": max_data}

    return procesed_data, remaining_data
```

A sample of the data returned by the microservice can be seen in Code 5.8.

Code 5.8 Data preprocessed by the max microservice.

```
{
    "dataType": "processed",
    "preprocessingOperation": "Max",
    "T": "15 min",
    "historical": ["ID12/01", "Max"],
    "data": [
      {"timestamp": 1614072354,
       "value": 76
      },
      {"timestamp": 1614073254,
       "value": 78
      },
      [...]
    ]
}
```

Code 5.9 shows the most relevant part of the *preprocessing service provider*. It offers a single REST endpoint, implemented for this specific *case* with the Django framework[3]. This endpoint receives a request via JSON, where the name of the microservice to be executed, the parameters with which to execute the microservice and the data to be preprocessed are indicated. As the microservice to be executed is not known at development time, the *service provider* loads at runtime the module that implements the required microservice functionality (in this particular scenario `Average` or `Max`). This standardised implementation allows requesting any microservice without knowing a priori its functionality, making runtime composition possible. By this means, the *case manager microservice* (frozen spot) will invoke the different *preprocessing services* (hot spot) as many times as necessary according to the *case* description.

Code 5.9 Preprocessing service provider in Python

[3] https://www.djangoproject.com/

```
def post(self, request):
    data = json.loads(request.body.decode("utf-8"))

    p_name = data.get('service_name')
    p_parameters = data.get('parameters')
    p_data = data.get('data')

    try:
        module = __import__(p_name)
        procesed_data, remaining_data = module.execute(p_parameters, p_data)
        data = {
            "procesed_data": procesed_data,
            "remaining_data": remaining_data,
        }
        return JsonResponse(data, status=201)

    except:
        return JsonResponse({'status':'false','message':'Service invocation error'}, status=500)
```

Finally, Figure 5.24 shows the main components of the *source* node. These include the *data manager* (frozen spot), which records the data received by the *measurement microservice* of the Polar H7 heart rate sensor according to the reporting frequency specified within the *configuration*. *Measurement microservice* can be considered a hot spot as it allows to extend ASTREA functionality by adding new sensors. Additionally, another relevant information such as the MAC of the sensor, necessary to connect to it via Bluetooth as well as sensor name or sensor ID, is as well specified within the *configuration* of the node. This information is provided and can be modified by the *case manager* according to the case specification (e.g., if there is a change in the *case* and the *propagation of adaptations and upgrades* mechanism is triggered). The *data manager* will properly format the gathered data and send it to the corresponding node according to the information provided by the *data forwarding* policy, if such a node is available. If not, it will store the data temporarily (data buffering).

With respect to the *intermediary* and *destination* nodes, they both share the same main components Figure 5.25. In both, the *preprocessing microservices* are deployed, since in situations where there is more than one *intermediary* node that gathers the data in a distributed way or that the *source* node has a direct connection with the *destination* node, the preprocessing operations will be carried out on the *destination* node. The latter is particularly relevant when the data to be preprocessed are not complete.

The *communication unit* is installed within all nodes to manage the sockets and connections between devices. The *case manager* will act as the orchestrator of the *preprocessing microser-*

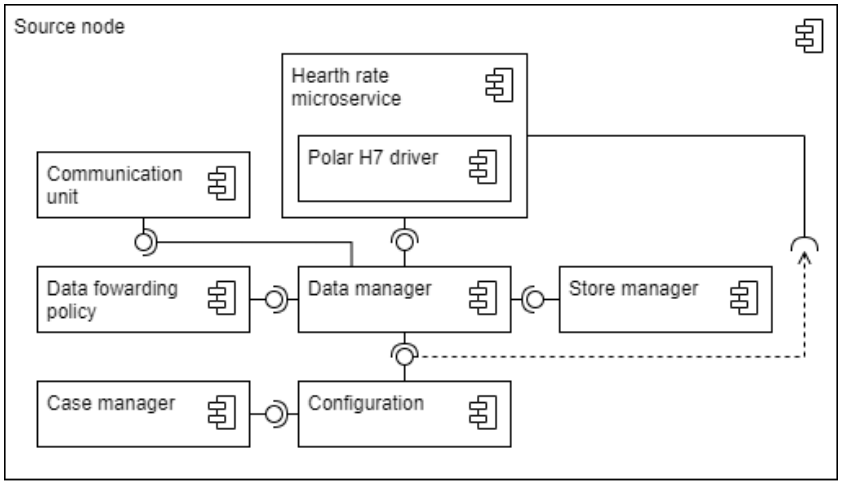


Fig. 5.24 *Component diagram of the source node.*

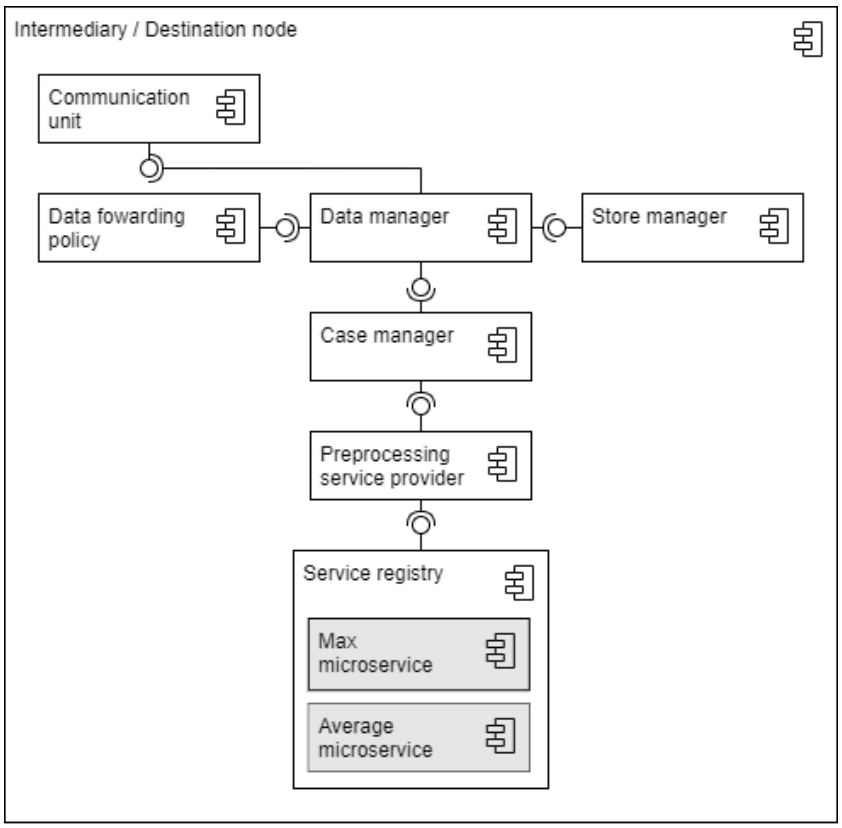


Fig. 5.25 *Component diagram of the intermediary and destination nodes.*

vices. It will request to the *data manager* the data to be preprocessed, and subsequently, it will request the *preprocessing service provider* for the appropriate microservices according to the *case* specification. This is possible by the standardised API with which the microservices offered are implemented.

5.14 Summary

This chapter describes ASTREA framework, as a novel solution to assist in the design and development of monitoring systems which can include sensors or wearables to collect context information. ASTREA supports the autonomous deployment and composition of monitoring systems to collect and gather user information (e.g., patient, residents, healthy control, etc.), environmental information from taking measurements of environmental factors, even the collection of data to determine the state of the system in order to improve its performance in terms of success rate, bandwidth and energy usage.

ASTREA includes the use of these devices (i.e., sensors, wearables, smartphones, etc.) which are envisaged, individually or as a grouping (i.e., node), which may be portable elements, and which may or may not be mobile. Within ASTREA, nodes are categorised by role based on their features. On these devices which are part of the network infrastructure, the first of the two mechanisms implemented in ASTREA, in particular, the *deployment, and propagation of adaptations and upgrades* mechanism deploys the system without human intervention. Adaptation or modification of systems once deployed to cover changes in the patient's condition, to mitigate problems arising from uncertainty or difficulty in predicting certain situations at the design phase, is feasible through the same mechanism that also provides for *adapting* and *upgrading*. In addition, the second mechanism, called *data gathering* mechanism transmits the collected data from the nodes where it originates to the nodes at the edge of the network where the information is centralised.

Solutions related to these three pillars bring together ASTREA framework. This includes ASTREA scope and what is involved. Specifically, it presents, as example, a motivation scenario in which it is feasible and of interest to deploy the proposal, lists the specific objectives to which ASTREA provides solutions, presents the system model in which the main entities are specified and explained, identifies the adaptation plans supported by ASTREA, details the *case* concept formalisation as one of the main foundations for specification of monitoring systems, presents ASTREA architecture design as well as the main services and microservices of the architecture and the dynamicity of certain components, the ASTREACE as visual editing tool developed for the creation of the monitoring systems (i.e., cases), the service repository which

stores the software pieces that can be reused to compose the systems, a case study describing the design, deployment and data gathering, and finally, an implementation and deployment to demonstrate the feasibility of ASTREA framework is described.

Part IV

Evaluation

Chapter 6

Evaluation

Chapter Abstract

This chapter includes materials and methods for the evaluation of ASTREA as well as the results obtained from two different studies. The first study includes the analysis of the time required in data propagation (simulating data transmission of a *case* specification and its microservices), and the success rate in data gathering, as well as the energy consumption of the nodes considering the sending of raw and already preprocessed data (i.e., in-network preprocessing). The second study also involves the evaluation of the time required for the data propagation (simulating *case* and microservices propagation) through the dynamic network, and the percentage of the success rate in data gathering. For the two studies, the inclusion of only monitoring actions versus monitoring actions together with preprocessing actions has been considered.

ASTREA has been evaluated using the ns-3 simulator and BonnMotion technologies [40]. In particular, each study applies different mobility models. The first study uses the Random Walk mobility model while the second uses the Reference Point Group Mobility (RPGM) and Manhattan Grid mobility models, approaching mobility patterns that could be found in a nursing home scenario. Heterogeneous nodes make up the network, belonging to different roles and with different capabilities.

The results reveal the importance of the network infrastructure but even more, the preprocessing actions to obtain useful information from raw data gathered. Furthermore, reducing the amount of data transmitted implies a reduction in transmission time and also reduces the energy consumption of the nodes, and thereby increasing the lifetime of the network.

Chapter Contents

6.1 Introduction

Evaluation of the ASTREA framework has primarily focused on data transmission and propagation time in a highly dynamic network under severe conditions because microservices dynamic deployment can not be achieved in the simulation environment. Nevertheless, the two mechanisms developed have been tested, *deployment, and propagation of adaptations and upgrades* and *data gathering* mechanisms. Destination nodes introduce modifications over a dynamic network topology, and in-network preprocessing is performed in the intermediary nodes. Therefore, propagation time, data gathering success rate and a comparison of the energy consumption of the node of transmitting raw data versus preprocessed data (useful information) have been studied.

To achieve this, two studies have been carried out. The propagation time indicates how much time is needed for the *case* subversions to reach all network nodes involved within it and data volume gathered reflects how much of the data generated at the source nodes reaches the destinations. Regarding the latter, the success rate of the gathered data is compared. Moreover, the first study considers data priority policies (established by the case designer during the *case* creation) and this also includes an analysis of the power consumption based on the data transmitted (raw data versus preprocessed data).

To this end, ns-3 simulator, version 3.21 and BonnMotion technologies have been used. Heterogeneous sensors have been simulated and different network configurations have been applied. In particular, varies the node role, number of each of them, and number of data generated per second. For the first study, 50 WSN (fixed nodes), a variable number of intermediaries, and 1 destination node compose the dynamic network infrastructure, while in the second study, 5 WSNs and 10 BSNs (mobile nodes) are additionally added, together with a variable number of intermediary and destination nodes.

The chapter is organised as follows. The objectives of each of the two studies and their respective network configuration details are described in Section 6.2. Moreover, in the Subsection 6.2.3 can be found a comparative summary of the two studies (objectives and network configuration details). Within each subsection information relating to study one and two can be found separately. In addition, evaluation results (Section 6.3) of propagation time (Subsection 6.3.1), data gathering success rate (Subsection 6.3.2) and energy consumption (Subsection 6.4) also comprise a discussion. Finally, Subsection 6.5 comprises a summary of the most relevant key points of the evaluation.

6.2 Materials and Methods

ASTREA framework is evaluated through the simulation of several scenarios and different network configurations using the ns-3 simulator, version 3.21, on Ubuntu 16.04 LTS 64-bit OS and the BonnMotion mobility scenario generation tool.

This section includes the details about the two studies which mainly have been designed with the objective of analysing data transmission and propagation time for *deployment, and propagation of adaptations and upgrades* mechanism, and volume of data gathered using *data gathering* mechanism considering data generated. The main differences between the two studies reside in the dynamic network topology configurations and the *case* specification.

The *cases* and network infrastructures which have been simulated seek to emulate real nursing homes similar to those described in Section 5.12. Furthermore, the simulation process has been designed to prevent the possible influence of any random factors and every possible configuration has been simulated 100 times with different seeds. A simulation time of 3 hours was established for each configuration.

6.2.1 First Study

In this first study, we analyse mainly three issues:

- How much time is required for data transmission, which simulates the propagation of a *case* and its respective associated microservices through a dynamic network topology. The propagation of the subversions of a *case* and associated microservices originates at the destinations and ends at the source nodes. Therefore, the transmission in the data flow to measure the propagation time is in this direction.

- How much data arrives at the destination nodes based on how much data is generated at the source nodes, considering the dynamicity of the network, i.e., the success rate percentage of data gathering. Here, the data transmission flow originates at the source nodes and ends at the destination nodes.

- What is the energy consumption or energy cost when transmitting raw data compared to the energy consumption when reducing the volume of data to be transmitted from the intermediaries to destinations. The latter, it would be useful information because in-network preprocessing has been performed within the network at intermediary nodes.

In the evaluation, the success rate percentage in data gathering considering the role and number of nodes used with respect their abilities to act as servers from two points of view is also examined:

- Only monitoring actions have been included, simulating their inclusion within the *case* specification (subversions). Then, raw data gathered are transmitted through the network. However, in this evaluation, a higher volume of data is generated in the sources compared to that generated with real sensors, in order to perform the evaluation under extreme conditions (i.e., not in the best situation of the scenario). In [220], the authors estimate the volume of data generated by sensors in a smart city. Specifically, they estimate that a temperature sensor sends 2112 bytes of data per day, a noise sensor sends 31680 bytes of data per day, and an air quality sensor sends 13824 bytes per day. We generate between 0 and 150 bytes per second, so the equivalent data volume generated daily by their noise sensor can be achieved in approximately 211 seconds in any of our scenarios.

- Monitoring and preprocessing (average, maximum and minimum in 15-minute periods) actions have been included, simulating their occurrence within *case* subversions. Here, we partition the raw data set to perform the preprocessing actions within the intermediary nodes in order to obtain useful information and minimise the data volume to be transmitted.

In the simulated scenario three data priorities have been considered, being priority 1 the maximum priority and priority 3 the minimum. For a better comparable evaluation, an equal amount of data is generated for each priority.

Regarding energy consumption, the nodes are equipped in the simulation with a ns3::LiIonEnergySource. Initially, the battery charge of the intermediary nodes and sources is set to a random value between 60% and 100% of their maximum battery capacity (2.45 mAh). It should be noted that although ns-3 offers energy consumption models for data transmission, it does not have a consumption model for CPU operations available. Therefore, to simulate the execution of preprocessing operations on the intermediary nodes, a battery consumption penalty has been introduced on these nodes.

Dynamic Network Configuration Details

A simulated scenario composed of 50 source nodes, 1 fixed destination node and a variable number of intermediary nodes, which vary from 1 to 10 has been created. Each node role possesses particular features (i.e., computational capabilities) regarding the storage, mobility,

measures to gather, data volume generated and connection range. Configuration details of the heterogeneous nodes that comprise the simulated dynamic network can be seen in Table 6.1.

Table 6.1 *Configuration of the role of nodes that compose the network simulated.*

	Sources	Intermediaries	Destinations
Storage	1 GB	4 GB	Unlimitied
Data generation	[0-150] B/s	None	None
Approx. connection range	3 m	10 m	35 m
Multi-hop connection	No	Yes	-

A simulation area/arena of 4800m^2 (60m x 80m) was created within which the nodes were deployed. The nodes require an initial position that has been determined by considering the node role:

- The source nodes within WSNs have been positioned in fixed random positions in each simulation, with an Euclidean distance greater than 25 m between them.
- The intermediary nodes have been positioned randomly and follow a Random Walk mobility model with a variable speed from 0.5 m/s to 2 m/s and random pauses of a maximum of 30 seconds (Table 6.2).
- The destination node has been placed in a fixed position in the centre of the area.

Table 6.2 *Random Walk mobility model configuration for intermediary nodes.*

	Random Walk mobility model
Node min. speed	0.5 m/s
Node max. speed	2 m/s
Node max. pause	30 s

6.2.2 Second Study

The second study analyses the propagation time required by the *deployment, and propagation of adaptations and upgrades* mechanism, and data gathering success rate reached with *data*

gathering mechanism. In particular, the percentage in data gathering considering the role and number of nodes used with respect their abilities to act as servers from two points of view:

- Only monitoring actions have been included, simulating their occurrence within the *case* subversions. Then, raw data are generated at the source nodes and transmitted through the network to the destination nodes.
- Monitoring and preprocessing (average, maximum and minimum in 15-minute periods) actions have been included, simulating their occurrence within the *case* subversions. Here, we partition the raw data set to perform the preprocessing actions within the intermediary nodes in order to obtain useful information and minimise the data volume to be transmitted.

However, this study differs from the previous one (first study of Subsection 6.2.1) with respect to the network configurations used. In particular, the evaluation has been carried out by varying the number of nodes of each role, mainly intermediaries or destinations, in order to analyse how this factor influences the propagation times and data gathering success rates. As for the volume of data generated, as in the previous study, it is also greater than that generated under normal conditions so that the conditions are extreme.

Dynamic Network Configuration Details

Configuration details of the heterogeneous nodes that comprise the simulated dynamic network can be seen in Table 6.3.

Table 6.3 *Configuration of the nodes role that compose the network simulated.*

	Sources		Intermediaries	Destinations
	WSNs	BSNs		
Storage	1 GB	1 GB	4 GB	Unlimitied
Mobility	Fixed	Manhattan RPGM	Manhattan RPGM	Fixed
Number of sensors	[1-5]	[1-7]	None	None
Data generation	[0-150] B/s	[0-210] B/s	None	None
Approx. connection range	3 m	3 m	10 m	35 m

In this study, we differentiate between WSNs and BSNs. Specifically, source nodes to monitor physiological (10 nodes) and environmental (5 nodes) conditions have been created respectively. It has been assumed that BSNs include a set from 1 to 7 sensors clustered which could be also wearables, and the WSNs include a set from 1 to 5 sensors clustered to facilitate the measurement of those conditions required by each *case*. Furthermore, a variable number of both intermediary nodes which vary from 0 to 6 where 0 indicates that there are no data carriers, and destination nodes which vary from 1 to 6 have been created. However, it is worth mention that at least 1 source and 1 destination node are required to perform the evaluation of the propagation time and success rate percentage in data gathering. In addition, each node role possesses particular computational capabilities (also detailed in Table 6.1) in terms of storage, mobility, measures to gather, data volume generated and connection range.

In this second study, two different mobility models have been applied:

- Reference Point Group Mobility (RPGM) model [109] intends to simulate an open space (e.g., a garden) where it is more common for users to walk in a group, and
- Manhattan Grid mobility model [228], in order to simulate horizontal and vertical indoor corridors within a building that users usually walk through individually.

A simulation area/arena of 4800 m^2 (60 m x 80 m) was created within which nodes were deployed. The nodes require an initial position that has been determined by considering the node role:

- The source nodes within BSNs and intermediary nodes have been positioned randomly according to the mobility model used. A mobility model has available and restricted positions determined by the trace within the area, therefore the nodes have been placed randomly but in unrestricted locations. Moreover, as is the *case* in real world scenarios mobility speed can vary.

  A variable speed from 0.5 m/s to 2 m/s for RPGM model, and a mean mobility speed of 1.5 m/s was established for Manhattan Grid mobility model with pauses of a maximum of 350 s for each scenario.

- The source nodes within WSNs have been positioned in fixed random positions in each simulation, with an Euclidean distance greater than 25 m between them.

- Destination nodes have been positioned at strategic locations in order to promoting both processes, propagation and data gathering. The strategic locations are determined by prioritising:

1. those locations where the influx of users is greatest; and
2. maximising the area covered.

Here, covering the area near the center of the scenarios is considered also as a priority for both mobility models as well as the crossings in the corridors for Manhattan Grid mobility model because the possibilities of the source and intermediary nodes entering into the scope range are greater. In Figure 6.1, note that the destinations have been positioned in the simulation according to the number of available destinations (indicated at the top left corner) for both mobility models.

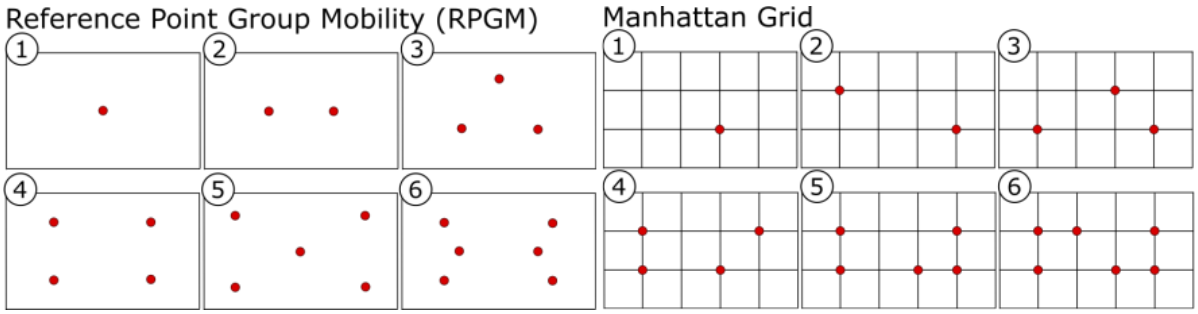


Fig. 6.1 *Position of the destination nodes within the scenarios according to the number of available destinations (indicated at the top left corner) for both mobility models.*

Details regarding those configuration parameters used to generate the mobility traces in BonnMotion are presented in Table 6.4.

6.2.3 Considerations of the First Study versus Second Study

In the Table 6.5, the main considerations of the two studies carried out have been listed. In the first study, energy consumption is analysed but not in the second study. In the first study also data priority has been taken into account but do not in the second study.

In the first study only fixed sensors are included, not mobile BSNs (source nodes). In the second study, BSNs are included, the number of which varies, as does the number of destinations nodes. The number of intermediary nodes varies in both studies. The node capabilities are the same for both studies although different mobility models are used. In the first one, the Random Walk mobility model has been used, while for the second one, the Reference Point Group Mobility and the Manhattan Grid mobility models were used. The three models can represent common movement behaviours in a nursing home.

Table 6.4 *RPGM model configuration for source nodes (BSNs) and Manhattan Grid mobility model for intermediary nodes.*

	RPGM model	Manhattan Grid mobility model
Node min. speed	0.5 m/s	0.5 m/s
Node mean speed	-	1.5 m/s
Node max. speed	2.0 m/s	-
Node max. pause	350 s	350 s
Group change probability	0.1	-
Group size standard deviation	2	-
Max. distance to group center	8 m	-
No. of blocks along x-axis	-	5
No. of blocks along y-axis	-	3
No. of groups	4	-
Pause probability	-	0.2
Scenario max. x	80 m	80 m
Scenario max. y	60 m	60 m
Speed change probability	-	0.4
Speed standard deviation	-	0.5
Turn probability	-	0.5
Update distance	-	10 m

6.3 Evaluation Results

This section shows the results obtained from the evaluation of ASTREA, considering the issues under study and the details in the configuration of the dynamic networks.

6.3.1 Propagation Time

Here, it can be found the results obtained from the two studies designed concerning the propagation time required by the *deployment, and propagation of adaptations and upgrades* mechanism.

First Study

Details of the configuration scenario can be found in Subsection 6.2.1. Here, it measures how long it takes for an adaptation or upgrade originating at the destination node (gateway) to reach 100% of the nodes in the network. Thereby, this provides an estimate of how long it would take to update the configuration (at the functional level) of the monitoring systems (*cases*).

Table 6.5 *Considerations of the first study VS those of the second study.*

	First study	Second study
Evaluation		
Propagation time	✔	✔
Success rate (%) in data gathering	✔	✔
Energy consumption	✔	✖
Monitoring actions	✔	✔
Preprocessing actions (in-network preprocessing)	✔	✔
Data priority	✔	✖
Network configuration		
WSNs (fixed source nodes)	✔[50 nodes]	✔[5 nodes]
BSNs (mobile source nodes)	✖	✔[10 nodes]
Intermediary (mobile source nodes)	✔[1-10 nodes]	✔[0-6 nodes]
Destination (fixed nodes)	✔[1 node]	✔[1-6 nodes]
Node capabilities		
Storage	≡	≡
Data generation	≡(WSN)/- (BSN)	≡(WSN)/ [0-210] B/s
Connection range	≡	≡
Simulation area	≡	≡
Simulation time/scenario	≡	≡
Mobility models		
Random Walk mobility model	✔	✖
Reference Point Group Mobility (RPGM) model	✖	✔
Manhattan Grid model	✖	✔

✔ means issue considered in the study.
✖ means issue does not considered in the study.
≡ means that the question is treated in both studies in an equivalent way (which is the same).

A time limit of 10 minutes has been set for the propagation of adaptations and upgrades. As can be seen in Table 6.6, in scenarios where there are only 1 or 2 intermediaries, the upgrades are not propagated to 100% of the nodes within the established time limit. However, we consider that more than 10 minutes may be an excessive delay because the system must be adapted or upgraded avoiding inconsistencies, so if this limit is exceeded, it would be advisable to incorporate more intermediaries. From networks with 5 intermediary nodes (Figure 6.2), the total propagation time drops below 18 seconds. From 8 intermediary nodes, the results obtained are similar. When 10 intermediary nodes are reached, the maximum time required for adaptations or upgrades is 13.33 seconds.

Table 6.6 *System deployment, adaptations and upgrades propagation times and maximum percentage completed for networks consisting of 50 sensor nodes, 1 destination node, and from 1 to 10 intermediary nodes. Random Walk mobility model.*

Number of intermediary nodes		1	2	3	4	5	6	7	8	9	10
% of network upgraded	25%	4.66s	5.09s	3.44 s	3.26 s	3.62 s	3.17s	2.75 s	3.09 s	3.18 s	2.78 s
	50%	278.51 s	8.81 s	4.76 s	5.15 s	4.66 s	4.88 s	4.02 s	4.29 s	4.27 s	4.18 s
	75%	-	-	8.86 s	11.44 s	6.40 s	6.18 s	5.24 s	5.39 s	5.12 s	5.31 s
	100%	-	-	253.95 s	73.03 s	17.51 s	16.69 s	17.96 s	14.58 s	14.04 s	13.33 s
% max. network reached, and system adapted or upgraded		69%	71%	100%	100%	100%	100%	100%	100%	100%	100%

The results obtained highlight the possibility of performing adaptations in the configuration of a runtime monitoring system on a dynamic and mobile-based WSN in a reasonable propagation time. Similarly, it may be feasible to transmit a specification to make upgrades to the monitoring system, which would lead to more noticeable changes in the system.

Second Study

Details of the configuration scenario can be found in Subsection 6.2.2. The results (Figure 6.3) show that the average propagation time decreases the greater the number of nodes because the probability in the number of interconnections is simultaneously increasing. As expected, propagation takes slightly less time to reach all nodes in the network when intermediary nodes move under the Manhattan Grid mobility model (Figure 6.3b). This is because under the RPGM model (Figure 6.3a), more than 1 intermediary node can move together, reducing the opportunities of propagation to other fixed nodes in the network. In contrast, with the Manhattan Grid model, the area available to move is reduced, making it relatively easy for nodes to meet each other.

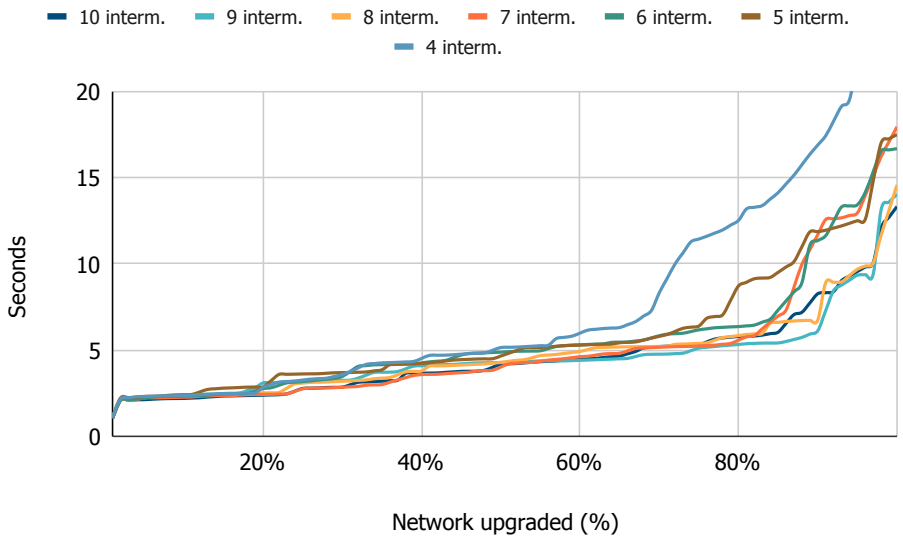
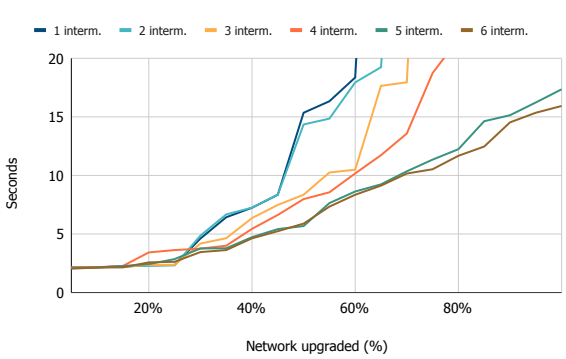
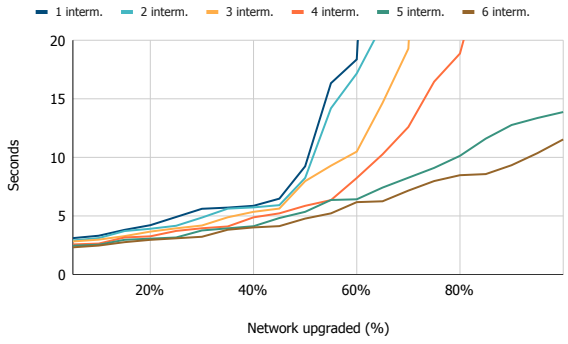


Fig. 6.2 *System deployment, adaptations and upgrades propagation times for networks consisting of 50 sensor nodes, 1 destination node, and from 4 to 10 intermediary nodes. Random Walk mobility model. Data in Table A.1.*



(a) *RPGM model. Data in Table A.2.*



(b) *Manhattan Grid mobility model. Data in Table A.3.*

Fig. 6.3 *Propagation results for networks composed of 5 fixed source nodes, 10 mobile source nodes, 1 destination node, and from 1 to 6 intermediary nodes.*

6.3.2 Data Gathering Success Rate

Here, it can be found the results obtained from the success rate of the data gathering analysed from the two studies designed.

First Study

As shown in Table 6.7, on average 11.64% more data is gathered of priority 1 than of priority 2, and 4.37% more of priority 2 than of priority 3.

Table 6.7 *Statistical results for data gathering (percentage gathered vs. generated) with and without prioritisation. No in-network preprocessing operations are performed on the data. Networks composed of 50 sensor nodes, 1 destination node, and from 1 to 10 intermediary nodes. Random Walk mobility model.*

	With priority			Without priority		
	Priority 1	Priority 2	Priority 3	Type 1	Type 2	Type 3
Max.	98.02%	80.20%	77.42%	89.22%	83.22%	82.72%
Min.	54.93%	40.79%	29.31%	36.68%	40.68%	39.68%
Avg.	78.48%	66.84%	62.47%	70.76%	67.96%	69.06%
Std. Dev.	0.12800	0.13155	0.15052	0.15528	0.14049	0.13040

Generally, this difference is observed for all the scenarios evaluated (Figure 6.4), highlighting how ASTREA manages to prioritise data delivery in accordance with what is specified in the *case*, gathering on average 78.48% of priority 1, 66.84% of priority 2, and 62.47% of priority 3. It should also be mentioned, that the maximum achieved gathering rates are 98.02% for priority 1, 80.20% for priority 2, and 77.42% for priority 3.

Data priority is of particular relevance in scenarios where the available hardware resources (bandwidth or storage) are low or where the contact of the source nodes with the intermediary nodes (data carriers) is spaced in time. Comparing these results with those of a *case* in which the data are not prioritised, as can be seen also in Figure 6.4 and Table 6.7, the data gathering of each type (without priority) is more balanced. Specifically, the difference in percentage of data gathered (without priority), on average, is fewer than 2.80% (Table 6.7), and these differences are mainly caused by random factors (e.g., positioning of the source node in the scenario).

In order to compare priority and non-priority data gathering, two identical tests (with the same scenario and nodes configuration) have been conducted. Nevertheless, it should be

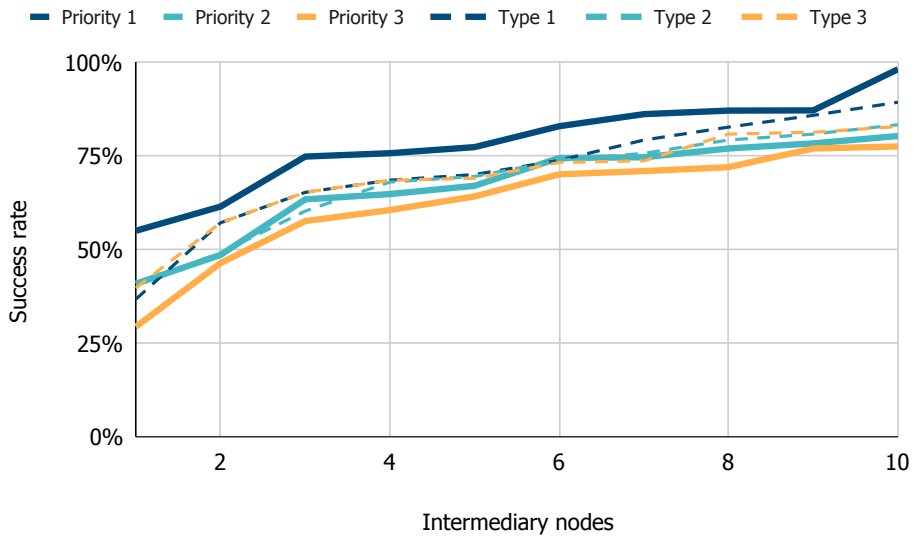


Fig. 6.4 *Data gathering with and without prioritisation (percentage gathered vs. generated). No in-network preprocessing operations are performed on the data. Networks composed of 50 sensor nodes, 1 destination node, and from 1 to 10 intermediary nodes. Random Walk mobility model. Data in Table A.4.*

clarified that for this comparison the generated data has been labelled with the priority although the prioritisation mechanisms for data transmission and storage have been disabled in the ASTREA framework.

The in-network preprocessing operations introduced in the simulation (average, maximum and minimum in 15-minute periods) reduce the total size of the final data (target information) by 80% (Table A.5). The target information is the information that should be obtained after applying the preprocessing operations. Therefore, when in-network preprocessing is applied, these operations are performed in the network itself, otherwise, they should be carried out in the gateway, cloud or data destination server. In the scenarios evaluated, in-network preprocessing is carried out at the intermediary nodes, therefore the proposed operations do not imply a direct reduction in the data transmission. There will be source nodes that transmit data directly to destination nodes, if they are within their range. Intermediary nodes that do not have the data corresponding to complete a time window (i.e., 15 minutes) to carry out the preprocessing operations will send the data to the destinations without preprocessing, after having transmitted the data they have already preprocessed.

Reducing the amount of data transmitted implies that the transmission time is reduced directly. Results in data gathering show that on average, 13.25% more data of the target information is gathered when preprocessing is applied in intermediary nodes (Table 6.8). The

Table 6.8 *Statistical results for data gathering when in-network preprocessing is applied vs. when it is not applied. Networks composed of 50 sensor nodes, 1 destination node, and from 1 to 10 intermediary nodes. Random Walk mobility model.*

	Without in-network preprocessing	With in-network preprocessing
Max.	80.72%	95.58%
Min.	41.68%	52.40%
Avg.	67.49%	80.74%
Std. Dev.	0.13101	0.15080

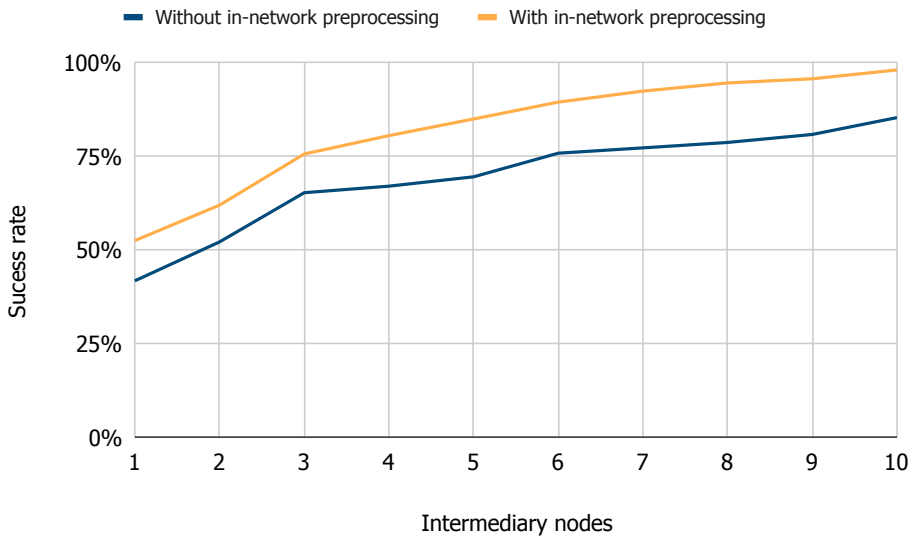


Fig. 6.5 *Data gathered percentage in relation to the data generated when in-network preprocessing is applied vs. when it is not applied. Random Walk mobility model. Data in Table A.5.*

difference between the two approaches becomes greater as the number of intermediary nodes in the network increases (Figure 6.5).

Second Study

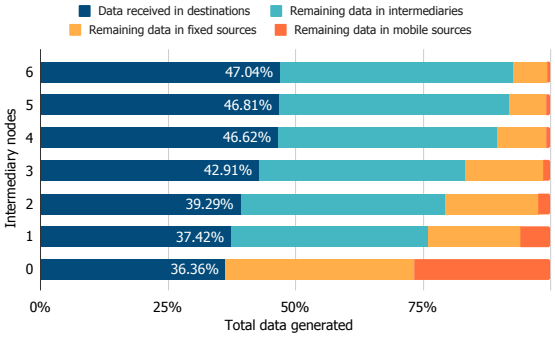
We analyse the success rate in data gathering from data received at destinations considering the volume of data produced at the sources. A comparison has been undertaken of the success rate obtained when the data gathered is raw data compared to when the preprocessing actions have been applied.

The network infrastructure deployed is one of the main factors influencing data receipt success rate together with the application of in-network preprocessing actions. As expected, the highest percentage of raw data was gathered when the maximum number of intermediary and destination nodes were deployed. In these circumstances, it was obtained an average success rate on the raw data gathering of the 88.24% for the RPGM model (Figure 6.6f), and of the 93.16% for the Manhattan Grid mobility model (Figure 6.7f). There are mainly two causes for not gathering data:

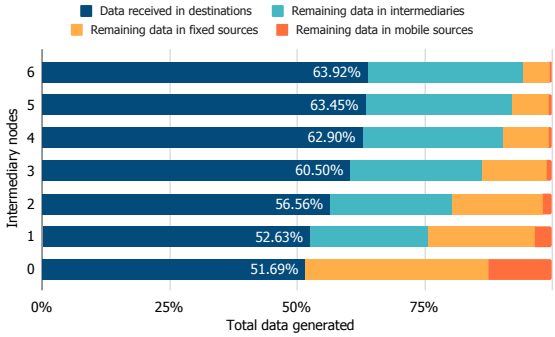
1. the simulation ends but some data generated in the sources did not reach the destinations;
2. an isolated source node is generating data, which are stored locally; or the connection time of a source node is insufficient to transmit a certain data volume, then when storage becomes saturated, the data is overwritten and cannot be retrieved. This does not happen in intermediary nodes because they cannot be elected as a server if they do not have free storage capacity so that the data they are transporting is not overwritten.

Next, we discuss the differences that arise in the success rate with respect to the number of available nodes of each role during the raw data gathering. When there are no intermediary nodes but the number of destination nodes increases from 1 to 6, the cumulative net increase in the success rate was 44.5% for the RPGM model, and 52.47% for the Manhattan Grid mobility model. This reveals that each time that a destination node is added to the scenario, there is an average increase in the success rate of 7.42% and 8.75% for each mobility model respectively.

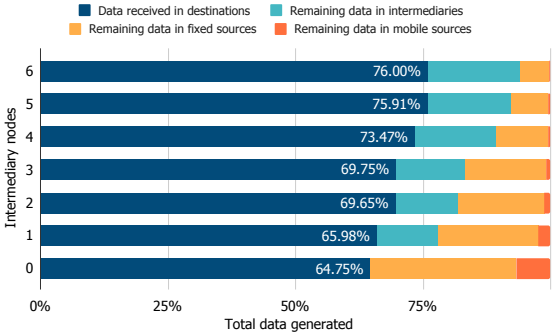
In both cases the increase of destination nodes follows a logarithmic growth. For the RPGM model the trend is determined by $f(x) = 0.356 + 0.25 \ln x$, where x is the number of destination nodes, with a coefficient of determination of $R^2 = 0.981$ (Figure 6.8a). For the Manhattan Grid mobility model the trend is determined by $0.275 + 0.294 \ln x$, where x is the number of destination nodes, with a coefficient of determination of $R^2 = 0.999$ (Figure 6.8b). The logarithmic trend reflects that the success rate percentage once a certain number of nodes have been deployed tends to stabilise.



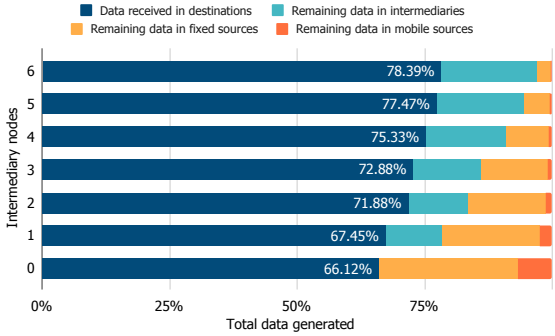
(a) *Networks composed of 1 destination node. Data in Table A.6.*



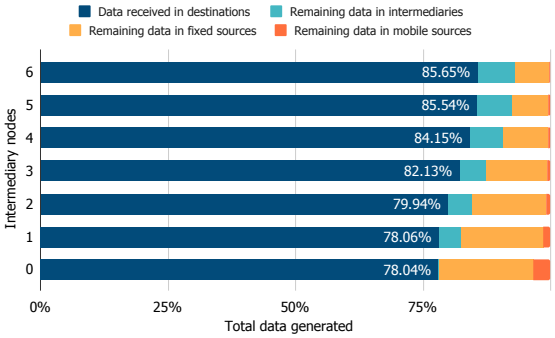
(b) *Networks composed of 2 destination nodes. Data in Table A.7.*



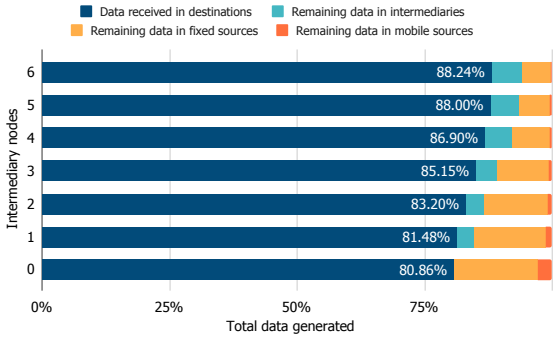
(c) *Networks composed of 3 destination nodes. Data in Table A.8.*



(d) *Networks composed of 4 destination nodes. Data in Table A.9.*

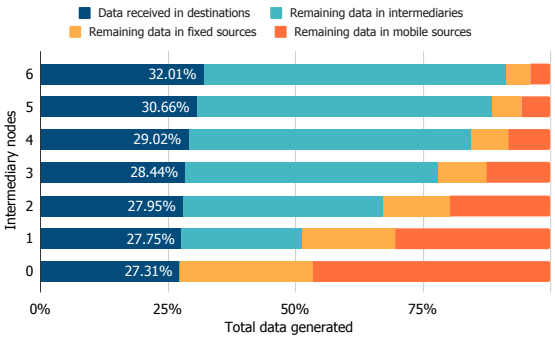


(e) *Networks composed of 5 destination nodes. Data in Table A.10.*

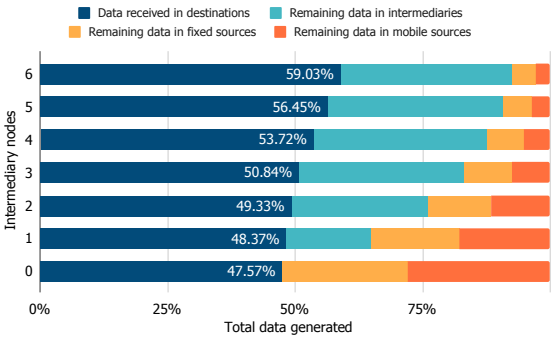


(f) *Networks composed of 6 destination nodes. Data in Table A.11.*

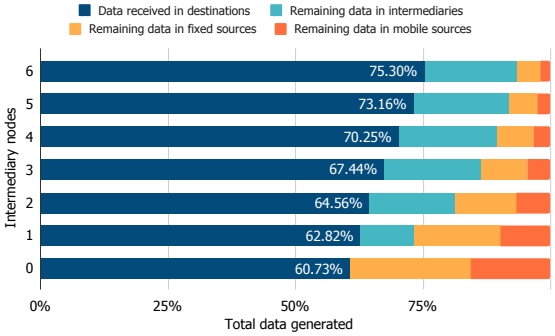
Fig. 6.6 *Data gathering results under the RPGM model. Networks composed of 5 fixed source nodes, 10 mobile source nodes, from 0 to 6 intermediary nodes, and from 1 to 6 destination nodes. No in-network processing applied.*



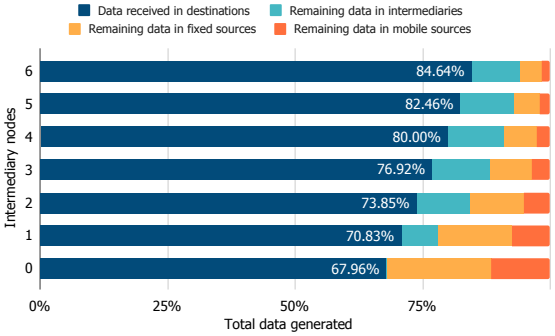
(a) *Networks composed of 1 destination node. Data in Table A.12.*



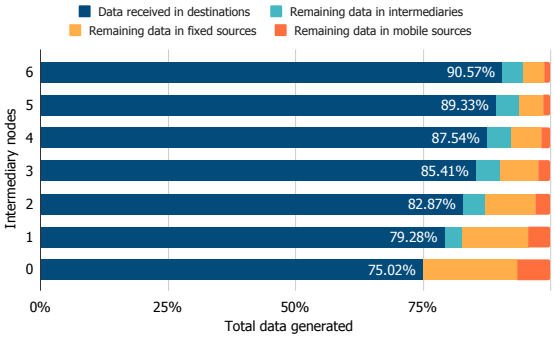
(b) *Networks composed of 2 destination nodes. Data in Table A.13.*



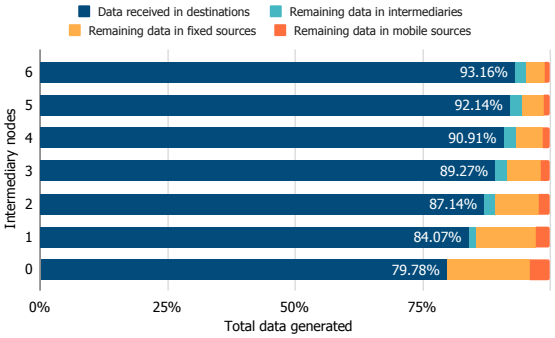
(c) *Networks composed of 3 destination nodes. Data in Table A.14.*



(d) *Networks composed of 4 destination nodes. Data in Table A.15.*

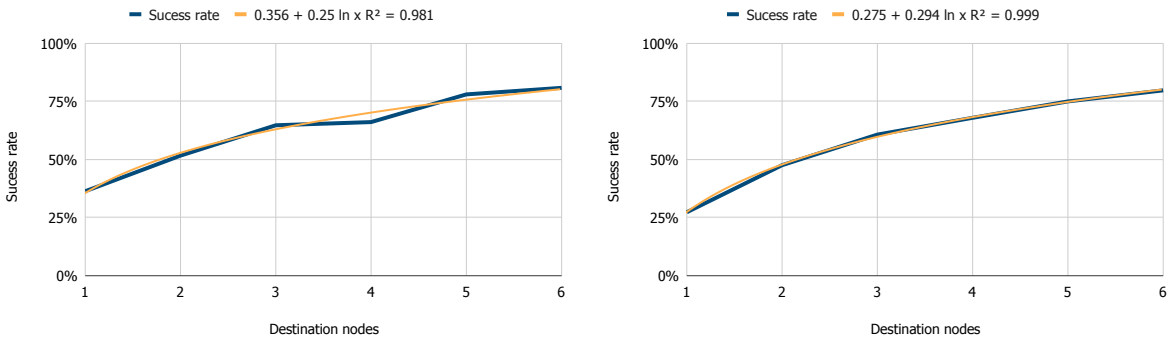


(e) *Networks composed of 5 destination nodes. Data in Table A.16.*



(f) *Networks composed of 6 destination nodes. Data in Table A.17.*

Fig. 6.7 *Data gathering results under the Manhattan Grid mobility model. Networks composed of 5 fixed source nodes, 10 mobile source nodes, from 0 to 6 intermediary nodes, and from 1 to 6 destination nodes. No in-network processing applied.*



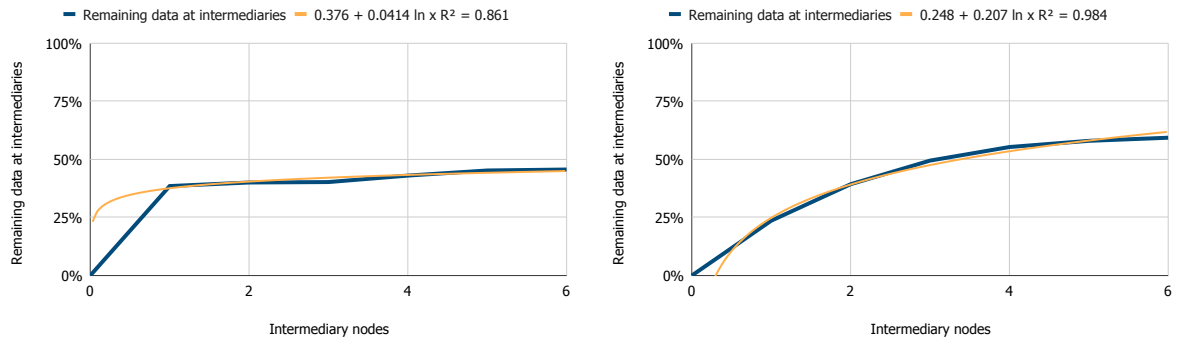
(a) *RPGM model.* **(b)** *Manhattan Grid mobility model.*

Fig. 6.8 *Data gathering results for networks composed of 5 fixed source nodes, 10 mobile source nodes, no intermediary nodes, and from 1 to 6 destination nodes. No in-network processing applied.*

It should be noted that the increase in the number of destination nodes does not always imply an increment in the percentage of data gathered because the fixed positions where the destination nodes are deployed have a great influence in the success rate result. Thus, a strategic deployment of destination nodes, in order, to cover the area, is decisive for a high success rate. In this regard, with configurations in which the number of destinations is maintained and the number of intermediaries varies, it can be noted that the increase in data gathered at the destination increases only slightly. In particular, a 10.71% increase for RPGM model (Figure 6.6a), and 4.7% for Manhattan Grid model (Figure 6.7a).

In contrast, it can be observed that the remaining data at the intermediary nodes is high. The remaining data in the intermediary nodes are those data collected from the source nodes which the intermediaries have not been able to retransmit to the destination node, either because they have not reached a destination node or because the time or bandwidth has not been enough to download all the data. For the RPGM model, intermediary nodes collect between 38.51% and 45.63% of the information generated (Figure 6.9a), while in the Manhattan Grid model it is between 23.60% and 59.32% (Figure 6.9b). As shown in Figure 6.9, the trend line is more pronounced for the Manhattan Grid mobility model than for the RPGM model. Therefore, adding a new intermediary node in the RPGM model has a lower impact than adding it in the Manhattan Grid mobility model. The reason is that in the Manhattan Grid mobility model, the nodes only move along a predefined path, i.e. the area of movement is much more restricted. In contrast, in the RPGM model, nodes move in groups, therefore, if in a group there are two intermediary nodes with the same movement pattern, they will visit the same source nodes, which will reduce their impact on data gathering. It is positive that the data remains within

the intermediaries rather than sources because the first ones will eventually have a destination within range to transfer the data.



(a) *RPGM model.* **(b)** *Manhattan Grid mobility model.*

Fig. 6.9 *Data gathering results for networks composed of 5 fixed source nodes, 10 mobile source nodes, 1 destination node, and from 0 to 6 intermediary nodes. No in-network processing applied.*

In this context, i.e. where no preprocessing operations are performed in the network, the fraction of data received by destination nodes through the intermediaries, either through multi-hop connections to source nodes or by carrying it, is small compared to that received through direct connections to source nodes within the scope of the destination itself. The maximum percentage (22.69%) delivered to the destination nodes via intermediary nodes occurs when there is only 1 destination and 6 intermediaries under the RPGM model (Figure 6.10a). For these configurations, when there is only 1 destination, the percentage of data reaching destinations via intermediary nodes, as would be expected, is higher than when there are more destinations because the 1-hop connections established between sources and destinations provide a direct data transmission. However, it was not expected that the results would be so similar with configurations of 3 and 4 destination nodes, and more specifically, when 5 intermediary nodes are operating (Figures 6.10c and 6.10d). The results obtained with 5 and 6 destination nodes are also very similar (Figures 6.10e and 6.10f). This is because the position of the nodes is similar when there are 3 or 4 destination nodes, and when there are 5 and 6 destination nodes. The coincidence, when there are 3 or 4 destination nodes and when there are 5 and 6 destination nodes respectively, also occurs with the data received at the destination nodes (Figure 6.6).

The data received at the destinations via the intermediary nodes with the Manhattan Grid mobility model (Figure 6.11), reached its maximum percentage (19.71%) for scenarios with 6 intermediary nodes and 4 destination nodes. The percentages achieved are similar when the scenarios include 2, 3 and 4 destination nodes (Figures 6.11b, 6.11c and 6.11d). The results

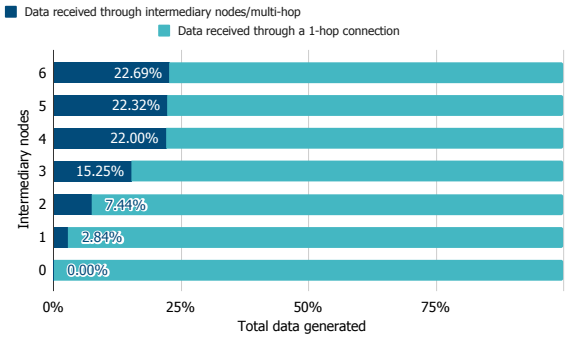
reveal that if a scenario includes 1 destination node in a central position, even if 6 intermediaries are incorporated, such intermediaries deliver a lower percentage of data at the destination node than when there are 2, 3, and 4 destinations, and significantly less than when using the RPGM mobility model. Surprisingly, the results obtained with 6 intermediary nodes, both with 1 and 6 destination nodes (Figures 6.11a and 6.11f), are very similar. Therefore, a strategic positioning of the destination nodes is essential. This also explains that from 5 destinations, the bottleneck becomes smaller, and therefore, including more destinations produces a greater benefit to direct connections to source nodes than to intermediary nodes.

Finally, it should be mentioned that the more destination nodes there are in the scenario, the less impact the intervention of the intermediary nodes has. However, intermediary nodes inclusion is not unimportant because the data gathering of 8.89% percent of the data for RPGM depends on them for scenarios with 5 destination nodes, and 17.18% percent of the data for the Manhattan Grid mobility model.

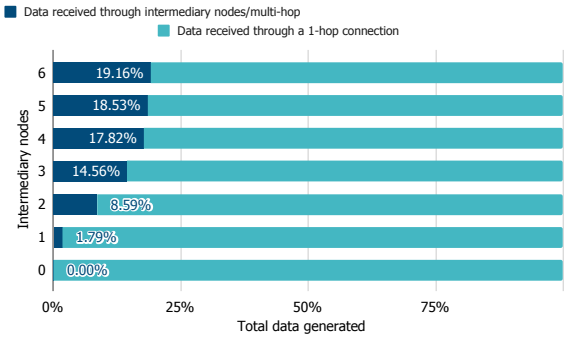
In the results obtained, it can be observed that there is a bottleneck in the offloading of data from the intermediaries to the destinations, in general, when the number of destination nodes is reduced. One way to address this problem, as proposed in this work, is to perform the preprocessing operations at the network nodes themselves, thus reducing the amount of data to be transmitted. Therefore, for the following scenarios, average, maximum and minimum preprocessing operations have been introduced at the intermediary nodes. It consists of arithmetic operations of the data in 15-minute periods, reducing the amount of data to be gathered by the destination nodes by approximately 80%. It should be noted that if the data does not pass through the intermediary nodes explicitly, i.e. it goes directly from the source nodes to the destination nodes, either through a direct connection or a multi-hop connection acting the intermediary nodes as routers in the connections, this data will not benefit from the preprocessing operations.

When there is in-network preprocessing as opposed to when there is not, in networks with 6 intermediary nodes and 6 destinations, it was obtained an improvement of 2.7% and 0.79% in data gathering with the RPGM (Figure 6.12f) and Manhattan Grid mobility (Figure 6.13f) models, respectively.

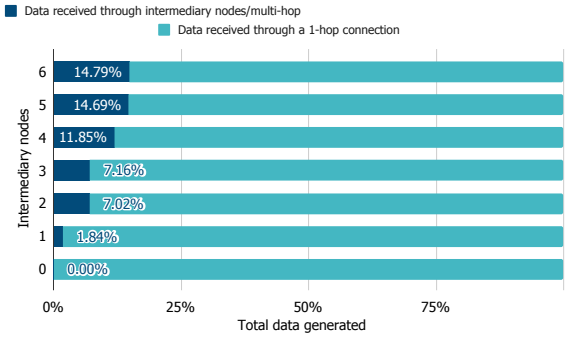
Furthermore, it can be seen that the remaining data at the intermediary nodes is particularly high and that for single-destination configurations, the remainder increases as the number of intermediaries increases for both mobility models (Figures 6.6, 6.7, 6.12 and 6.13). This is because the bottleneck occurs at each intermediary independently of the others and because the probability of a connection being established between each intermediary and the destination is lower than when there are more destinations. As the number of destinations increases,



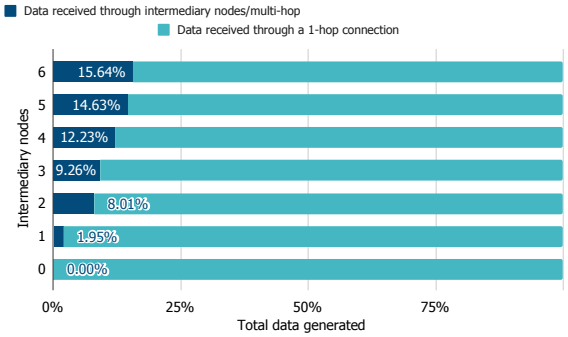
(a) *Networks composed of 1 destination node. Data from Table A.6.*



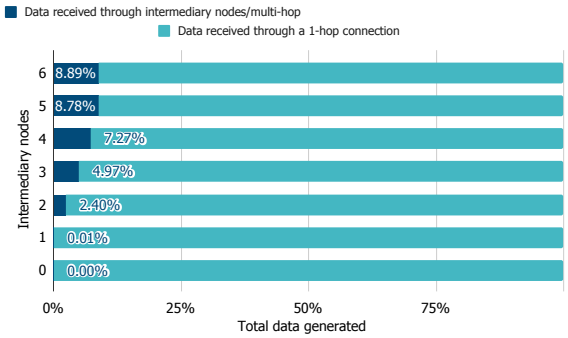
(b) *Networks composed of 2 destination nodes. Data from Table A.7.*



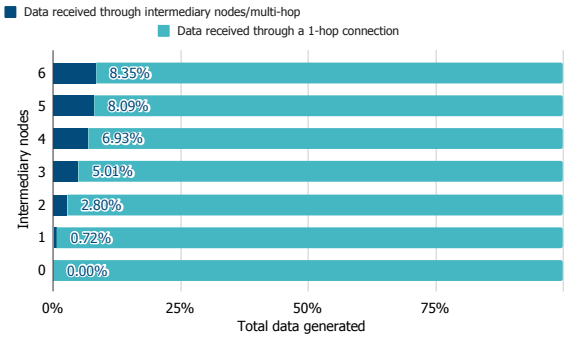
(c) *Networks composed of 3 destination nodes. Data from Table A.8.*



(d) *Networks composed of 4 destination nodes. Data from Table A.9.*

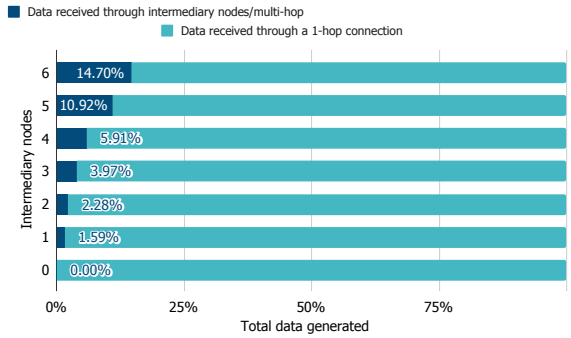


(e) *Networks composed of 5 destination nodes. Data from Table A.10.*

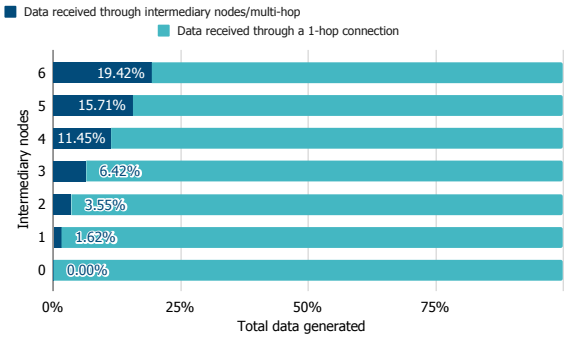


(f) *Networks composed of 6 destination nodes. Data from Table A.11.*

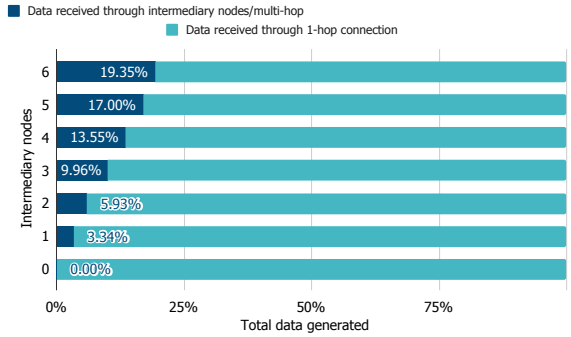
Fig. 6.10 *Data gathering results under RPGM model considering 1-hop and multi-hop connections. Networks composed of 10 mobile source nodes, 5 fixed source nodes, from 0 to 6 intermediary nodes, and from 1 to 6 destination nodes. Without in-network preprocessing operations.*



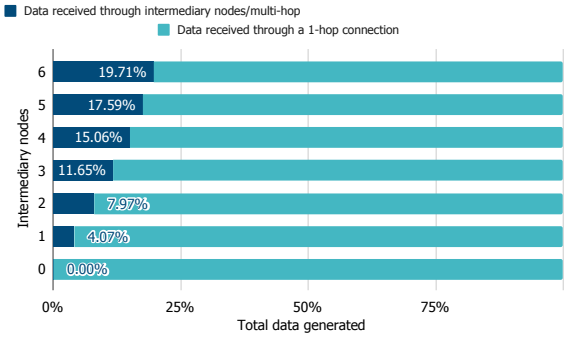
(a) *Networks composed of 1 destination node. Data from Table A.12.*



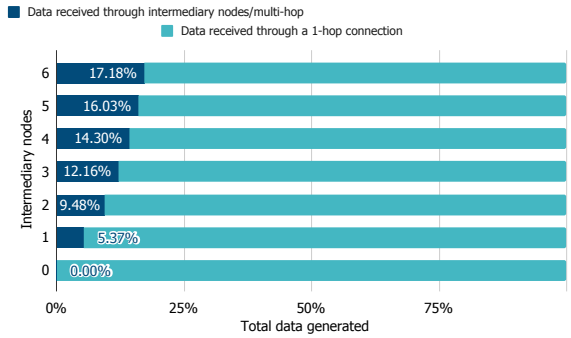
(b) *Networks composed of 2 destination nodes. Data from Table A.13.*



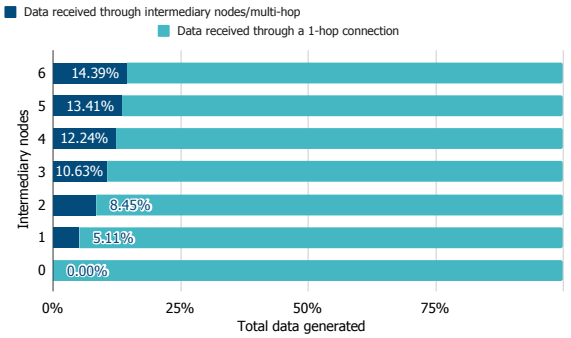
(c) *Networks composed of 3 destination nodes. Data from Table A.14.*



(d) *Networks composed of 4 destination nodes. Data from Table A.15.*

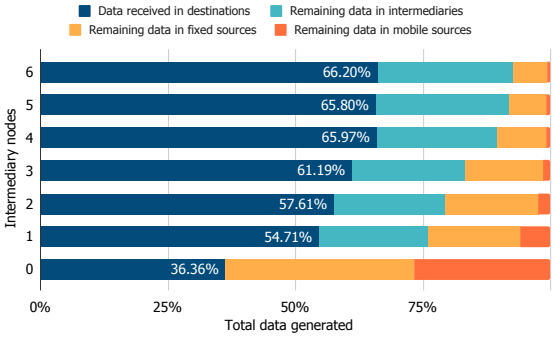


(e) *Networks composed of 5 destination nodes. Data from Table A.16.*

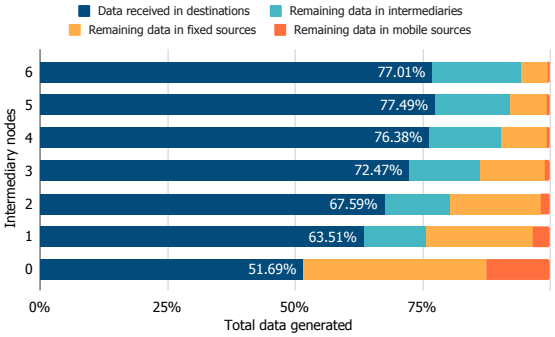


(f) *Networks composed of 6 destination nodes. Data from Table A.17.*

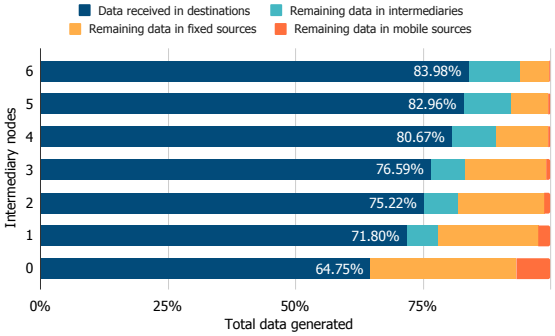
Fig. 6.11 *Data gathering results under Manhattan Grid mobility model considering 1-hop and multi-hop connections. Networks composed of 10 mobile source nodes, 5 fixed source nodes, from 0 to 6 intermediary nodes, and from 1 to 6 destination nodes. Without in-network preprocessing operations.*



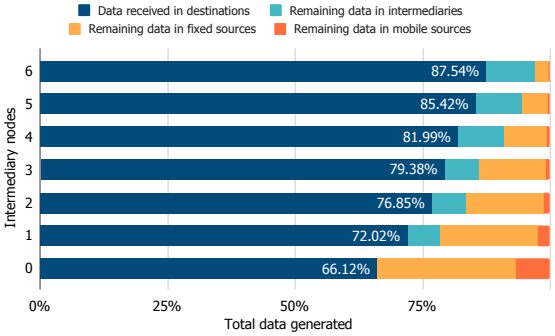
(a) *Networks composed of 1 destination node. Data in Table A.18.*



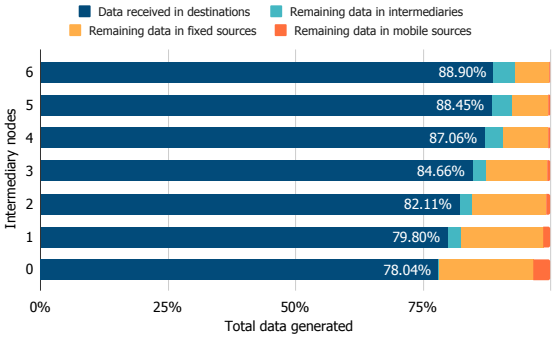
(b) *Networks composed of 2 destination nodes. Data in Table A.19.*



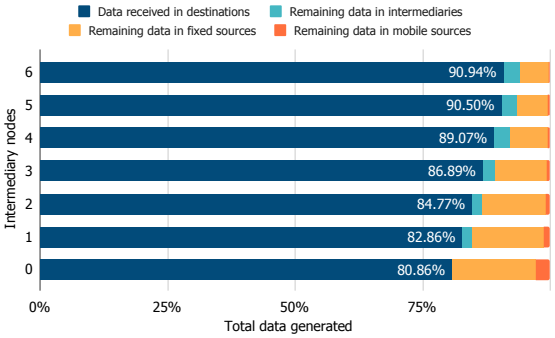
(c) *Networks composed of 3 destination nodes. Data in Table A.20.*



(d) *Networks composed of 4 destination nodes. Data in Table A.21.*

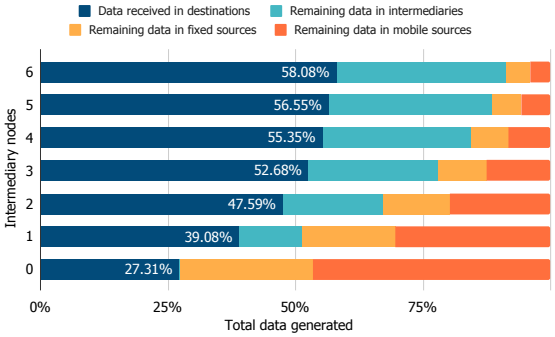


(e) *Networks composed of 5 destination nodes. Data in Table A.22.*

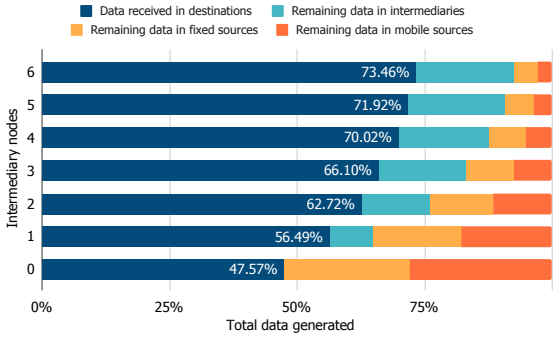


(f) *Networks composed of 6 destination nodes. Data in Table A.23.*

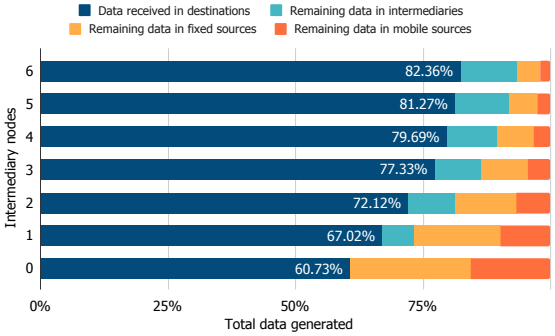
Fig. 6.12 *Data gathering results under RPGM model. Networks composed of 5 fixed source nodes, 10 mobile source nodes, 0 to 6 intermediary nodes and 1 to 6 destination nodes. With in-network preprocessing operations.*



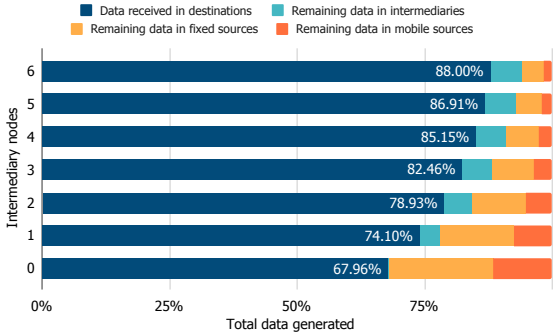
(a) *Networks composed of 1 destination node. Data in Table A.24.*



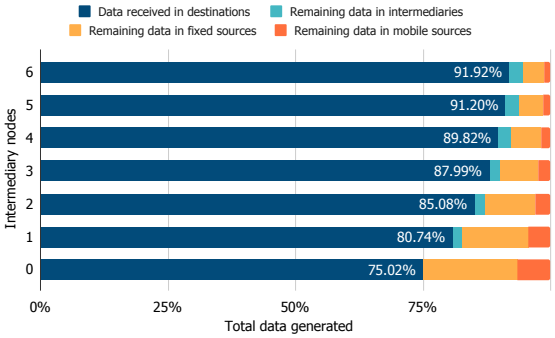
(b) *Networks composed of 2 destination nodes. Data in Table A.25.*



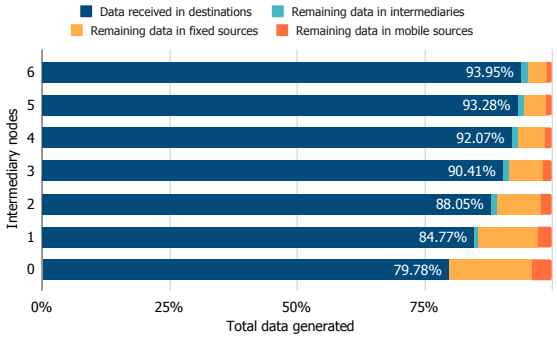
(c) *Networks composed of 3 destination nodes. Data in Table A.26.*



(d) *Networks composed of 4 destination nodes. Data in Table A.27.*



(e) *Networks composed of 5 destination nodes. Data in Table A.28.*



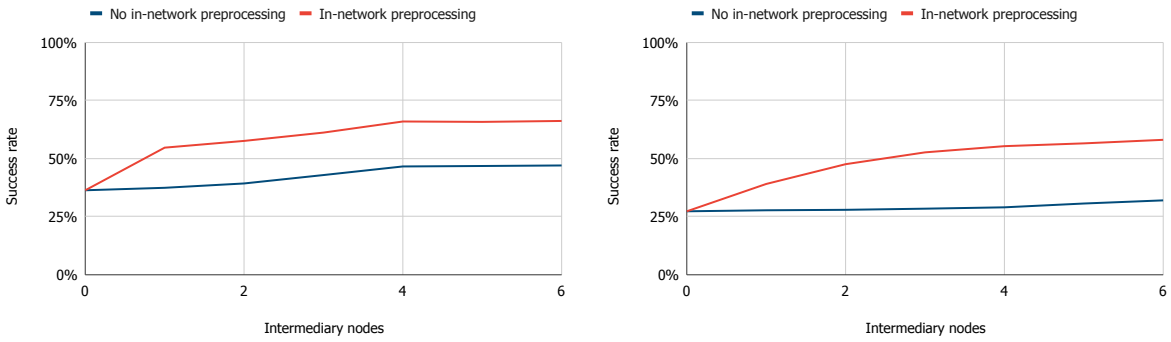
(f) *Networks composed of 6 destination nodes. Data in Table A.29.*

Fig. 6.13 *Data gathering results under Manhattan Grid mobility model. Networks composed of 5 fixed source nodes, 10 mobile source nodes, 0 to 6 intermediary nodes and 1 to 6 destination nodes. With in-network preprocessing operations.*

the volume of data remaining at the intermediary nodes decreases because the probability of establishing direct connections between the source and destination nodes also increases, which reflects that the bottleneck is being alleviated. The bottleneck is also clearly alleviated by performing in-network operations.

In addition, when there is only a single destination node and 6 intermediaries, and in-network preprocessing is applied, the destinations received 66.20% of the data with the RPGM, and 58.08% of the data with the Manhattan Grid mobility models while destinations received 47.04% and 32.01% respectively when no preprocessing operations are performed. Therefore, the results show that by performing in-network preprocessing a larger volume of data is gathered, which in turn is useful information. In parallel, it should be noted that in these configurations mobile source nodes have been included, and that when these mobile source nodes establish connection with a destination node, they directly download the data via direct connection, for this reason, also the volume of data received at the destination nodes via a one-hop connection, is higher.

Focusing now on when network preprocessing is applied. In this evaluation, in-network operations are performed within the intermediary nodes, therefore, when there are no intermediary nodes but the number of destination nodes increases from 1 to 6, the results obtained are identical to those obtained when no preprocessing operations are applied within the network.



(a) *RPGM model.* **(b)** *Manhattan Grid mobility model.*

Fig. 6.14 *Data gathering results with and without in-network processing for networks composed of 5 fixed source nodes, 10 mobile source nodes, 1 destination node, and from 0 to 6 intermediary nodes.*

It should be noted that when there is only 1 destination node and the number of intermediary nodes varies, it can be observed that the increase in data gathered at the destination increases considerably. In particular for these configurations, when in-network preprocessing is applied and intermediary nodes increase from 1 to 6, the success rate is increased by 29.84% for

the RPGM model (Figure 6.14a) and by 30.77% for the Manhattan Grid mobility model (Figure 6.14b). In contrast, when no preprocessing is applied, the increase in data gathering success rate is 4.7% for the RPGM model and 10.71% for the Manhattan Grid mobility model.

The results also reveal how the preprocessing of the data considerably increases the effectiveness of the intermediary nodes, increasing the amount of data gathered in the destination nodes through them. In particular, 45.07% (Figure 6.15a) and 52.98% (Figure 6.16a) of the data are delivered to the destinations for networks composed of a single-destination node and 6 intermediary nodes with the RPGM and Manhattan Grid mobility models, respectively. It can also be seen in Figures 6.15 and 6.16 that the data received through intermediary nodes always decreases as the number of destination nodes increases, and that no peaks and troughs occur because network preprocessing avoids bottlenecks.

Finally, the following reflections can be summarised. Adding new destination nodes when doing in-network preprocessing benefits direct connections more than intermediary nodes, because the preprocessed data minimises the occurrence of bottlenecks. When no in-network preprocessing is performed, the inclusion of destination nodes benefits both direct connections and indirect connections via intermediary nodes.

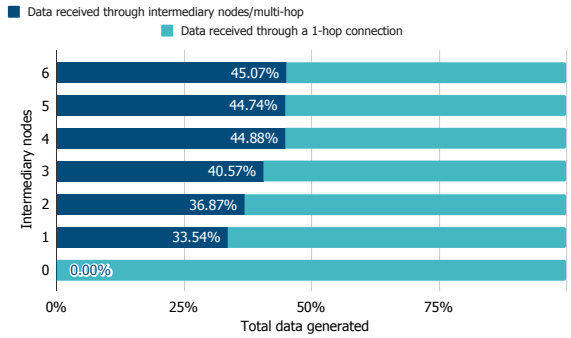
6.4 Energy Consumption

The results shown in the Figure 6.17 concerning to energy consumption reflect that raw data transmission has a high cost in the energy consumption of the nodes. In particular, here, the network lifetime is 2.92 hours but introducing preprocessing operations, the lifetime of the network increases by 2.5 hours, i.e., up to 5.42 hours in total.

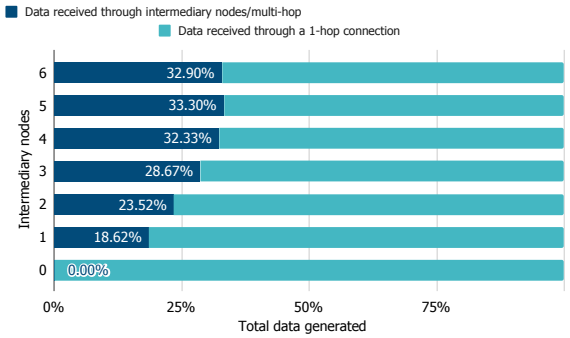
Therefore, reducing the amount of data transmitted implies a reduction in the power consumption of the nodes, and thereby an increase in the lifetime of the network.

6.5 Summary

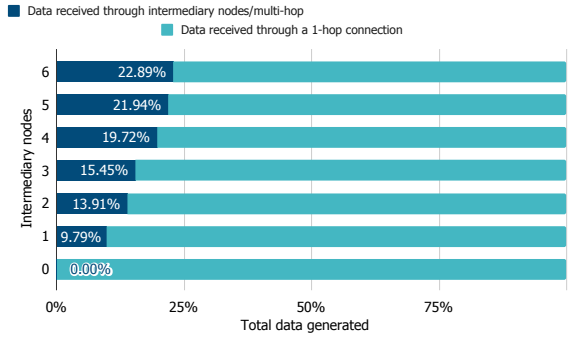
In this chapter, the ASTREA framework has been evaluated using the ns-3 simulator. The scenarios evaluated are intended to be similar to those in which ASTREA can be useful, such as a nursing home. ASTREA has been evaluated under three different mobility models, Random Walk mobility model, RPGM model, and Manhattan Grid mobility model, as well as with a set of roles of heterogeneous nodes. The main issues to be evaluated have been data transmission and propagation time simulating the distribution of the case specification and associated microservices, and quantification of the data gathered at the destinations as well as



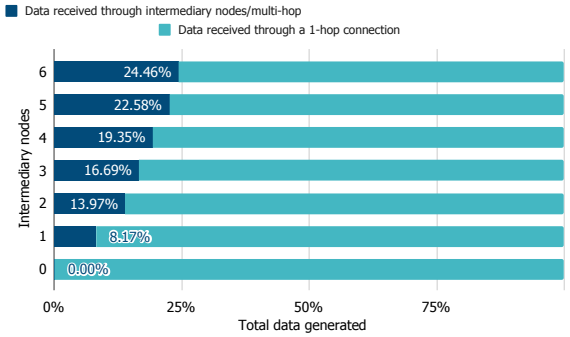
(a) *Networks composed of 1 destination node. Data in Table A.18.*



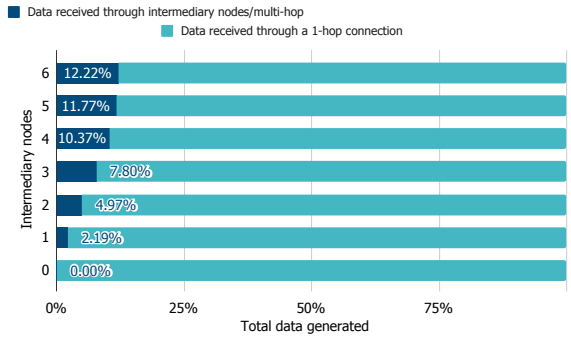
(b) *Networks composed of 2 destination nodes. Data in Table A.19.*



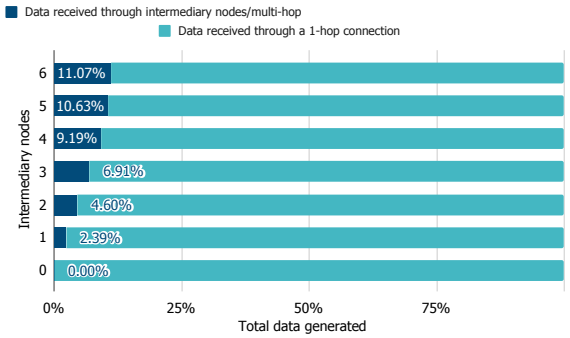
(c) *Networks composed of 3 destination nodes. Data in Table A.20.*



(d) *Networks composed of 4 destination nodes. Data in Table A.21.*

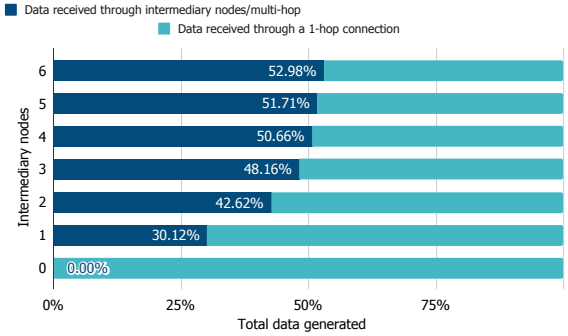


(e) *Networks composed of 5 destination nodes. Data in Table A.22.*

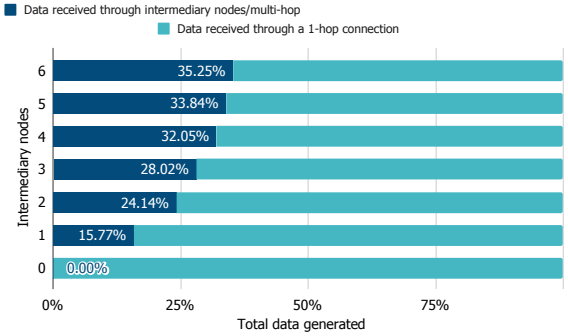


(f) *Networks composed of 6 destination nodes. Data in Table A.23.*

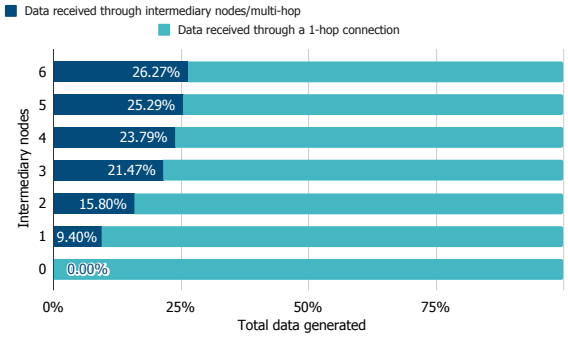
Fig. 6.15 *Data gathering results under RPGM model considering 1-hop and multi-hop connections. Networks composed of 10 mobile source nodes, 5 fixed source nodes, from 0 to 6 intermediary nodes, and from 1 to 6 destination nodes. With in-network preprocessing operations.*



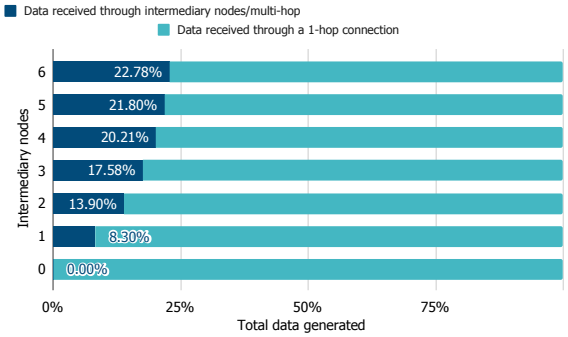
(a) *Networks composed of 1 destination node. Data in Table A.24.*



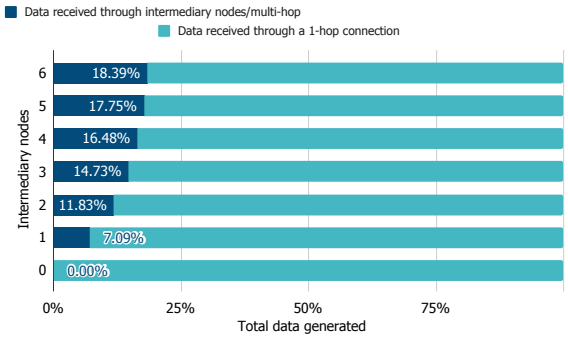
(b) *Networks composed of 2 destination nodes. Data in Table A.25.*



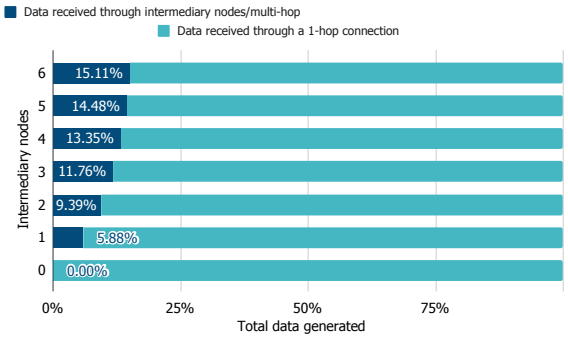
(c) *Networks composed of 3 destination nodes. Data in Table A.26.*



(d) *Networks composed of 4 destination nodes. Data in Table A.27.*



(e) *Networks composed of 5 destination nodes. Data in Table A.28.*



(f) *Networks composed of 6 destination nodes. Data in Table A.29.*

Fig. 6.16 *Data gathering results under Manhattan Grid mobility model considering 1-hop and multi-hop connections. Networks composed of 10 mobile source nodes, 5 fixed source nodes, from 0 to 6 intermediary nodes, and from 1 to 6 destination nodes. With in-network preprocessing operations.*

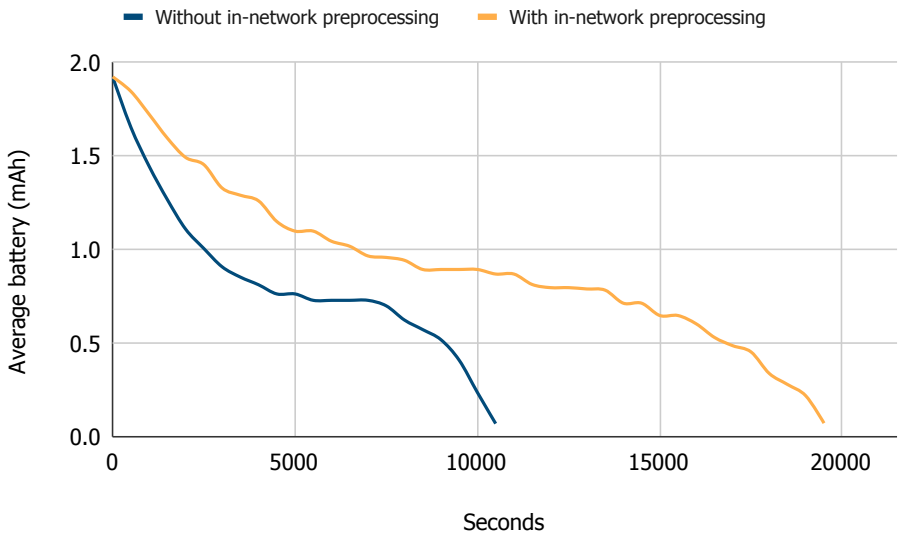


Fig. 6.17 *Average battery level of network nodes over time when in-network preprocessing is applied vs. when it is not applied. Data in Table A.30.*

the remnants within intermediary nodes. Moreover, energy consumption with respect to data transmission has been measured.

In data gathering, it has been considered that the gathering of certain measures may take priority over the gathering of others. Therefore, this issue has been compared with another specification where the gathering of measurements has the same priority. It has also been assessed how important it is to include in-network operations at intermediary nodes in order to obtain useful information as soon as possible and thus minimise the volume of data to be transmitted.

Regarding the propagation of adaptations or upgrades for the monitoring system configuration at runtime, it has been verified that when the number of intermediary nodes reaches a minimum of 5, for the scenarios studied, the monitoring system adaptation or upgrade introduced from a destination reaches all the nodes in the network in less than 18 seconds, regardless of the mobility model. This is an acceptable time to propagate a change in the configuration of a monitoring system in an ad-hoc WSN. However, it should be noted that the number of intermediary nodes required to provide an assumable propagation time will depend on the specific scenario, particularly, in its size and whether the movement pattern of the intermediary nodes can be targeted for this purpose.

Regarding data gathering, the inclusion of more intermediary nodes, per se, to carry out the data gathered from source nodes, if the data volume is large, does not imply a significant

increase in the data gathered at the destination nodes. This is mainly because there is a bottleneck between the intermediary nodes and the destination nodes at the time of offloading the data. For this reason, a large amount of data remains at the intermediary nodes and, although in a finite time this data will be transmitted to the destination node, the delay in its collection may invalidate the monitoring system. As a solution to this, the inclusion of priorities in the data to be gathered and transmitted helps to increase the success rate on priority data when the network is overloaded. However, the most influential improvement is found when preprocessing operations are introduced at the intermediary nodes themselves, which considerably reduces the data to be transmitted to the destination node and increases the gathering of relevant or useful data for the monitoring system. Moreover, the reduction of the data to be transmitted leads to a direct improvement in the network's functional lifetime, as the nodes have their energy consumption reduced considerably.

Part V

Conclusions and Future Work

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this thesis work, the ASTREA framework is proposed as a comprehensive solution to develop adaptive and extensible monitoring systems from composition and reuse of software. These monitoring systems must support adaptation and upgrades post-deployment at runtime, and data gathering. In addition, these systems have to be operational and must be deployed on a dynamic and mobile network infrastructure. To achieve this, a study of fundamentals, technologies, and research work related to our main objective and the goals defined has been carried out. Furthermore, technical issues of software development, hardware devices, and networks have also been also analysed.

The SLE, fibromyalgia, and SGA foetuses and low birth weight babies application domains have been studied during the development of this thesis. In addition, environmental monitoring has been considered as part of context information. Two monitoring systems have been developed for such domains, in particular they have been applied to SGA and to a study of sleep quality assessment in people suffering from SLE. The systems requirements have been defined with the collaboration of experts (psychologists, doctors, nurses, midwives, educators and therapists). To monitor environmental context, we developed a mHealth system that integrate CEnMO application to take luminosity, environmental temperature, noise, humidity and pressure measurements from a mobile device with built-in sensors. Data collected with CEnMO and physiological data collected with an actigraph are objective information because the collection of these measurements does not require human intervention. This objective information was combined with information from standardised questionnaires (subjective information) to conduct the sleep quality study. In this case, environmental monitoring helped

to check the ecological validity of the data and to study how environmental factors affect sleep quality.

Subsequently, we developed mPOP, another mHealth system to provide a psycho-educational programme for pregnant women, and their partners, with low weight-for-gestational-age foetuses. These programmes are offered through three mobile applications (VIVEmbarazo, VIVECrianza Hospitalización and VIVECrianza No Hospitalización). The psychoeducational programmes have been developed by experts in the field, and can be customised according to the evolution of foetuses, babies and their parents. Parent information is also managed by a domain expert through an authoring tool. All this is supported by AdaptEn module which mainly comprises the set of services that are orchestrated to load and to allow easy modification of programmes, as well as to ensure information security.

The application domains addressed have been useful to identify common characteristics, similarities, requirements, software solutions and hardware devices. All of them have helped us to acquire the knowledge and basis for proposing and evaluating new ideas, which have been the starting point for the core of the proposal, the ASTREA framework. ASTREA has been created to assist domain-specific software developers in the design and development of monitoring systems that include sensors and wearables to monitor context information (physiological variables and environmental factors). ASTREA provides solutions to a number of common problems, requirements and goals of modern monitoring systems, encouraging the reuse of software and automating certain tasks that in most of the systems need to be carried out such as the deployment, adaptation and upgrades of the systems, and data gathering.

Within ASTREA, the monitoring systems definition are specified as individual *cases* and they can be designed by a designer that could be a domain-specific software developer but who does not need to have technical or programming skills. From the *case* concept, it is possible to define the functionality of the monitoring systems, the devices (sensors and wearables) to be used, to establish links between functionality and devices, and set certain configuration parameters. These are elements that are available in the ASTREACE visual editing tool, developed to facilitate the design process. The case designer can select an element, and drag and drop it on a canvas to be set. The final result will be a flowchart defining the *case* specification in JSON format.

Monitoring systems developed with ASTREA can operate in non-mobile and non-changing environments, but the challenge is just the opposite. With ASTREA, we intent to provide solutions to propagation of adaptations and upgrades at runtime for distributed monitoring systems already operational which are deployed within dynamic and mobile networks. Therefore, the *case subversion service* generates different subversions from the *case* specification, and accord-

ing to the capabilities of each device (or node) that can be part of the network infrastructure. In ASTREA, nodes are categorised within three different roles (sources, intermediaries and destinations). In particular, *sources* collect data; *intermediaries*, usually with higher capabilities than source nodes, are data carriers and can perform in-network operations; and *destinations* are the nodes with the highest capabilities of the infrastructure, that they act as a gateway between the Internet or the Cloud, being the entry point for the deployment of the case.

Once the system has been designed, it must be deployed on the devices, i.e., the microservices must be installed and the configuration must be set up. These tasks become more tedious, the greater the number of devices are involved and the greater the number of monitoring systems to be deployed. Furthermore, adaptations and upgrades are necessary because user needs can change, or because system needs to be extended, or even context changes. Context changes may imply self-adaptation in the monitoring system which is managed by plans defined within the *case* specification. In addition, ASTREA includes the *deployment, and propagation of adaptations and upgrades* mechanism which supports the autonomous deployment of the system, so domain-specific software developers must specify what they consider that need to be changed, and case specification is propagated through the dynamic network infrastructure, together with the associated microservices that will compose the monitoring system autonomously and at runtime, without the need to recompile any of its components. These microservices must have been previously developed by a software developer, and are stored in a repository for code mobility. The propagation of adaptations and upgrades starts at the destination nodes and ends at the source nodes. Here, we use MEs to propagate the adaptations and upgrades to nodes in the network that are not directly in range of the destination nodes.

The data gathering still presents performance challenges in terms of data gathering success rate, bandwidth and energy consumption. ASTREA implements the *data gathering* mechanism to perform in-network preprocessing operations to obtain useful information from raw data in order to minimise the volume of data to be transmitted as early as possible. Furthermore, we take advantage of the MEs to act as data carriers and to perform in-network preprocessing on the data available to them. In addition, the *data gathering* mechanism includes data prioritisation, data buffering, and data forwarding policy, thus adaptation here is also present.

Monitoring System (ASTREAMS) is the core of the developed mechanisms. ASTREAMS integrates several key microservices among which stand out an orchestrator (the case manager), and data and preprocessing managers. These microservices are responsible for coordination, case composition, data management and for performing in-network operations, if the node capabilities allows it. Also noteworthy is the data forwarding policy that is responsible for

electing the more suitable higher role node to which the data are sent, when there are several alternatives.

A motivating scenario has been described to illustrate the behaviour of users and caregivers in a nursing home where ASTREA can be useful. Users have BSNs to collect physiological information, there are also wireless sensors at specific points to collect measurements of the environmental factors, and caregivers carry mobile devices that act as carriers of data from the source nodes to the nodes where the information is centralised (destination nodes) with higher computational capabilities, located at the edge of the network (gateway). Each resident may require a specific monitoring system, which may even vary significantly depending on their health status. ASTREA framework covers both autonomous deployment of the case (monitoring system) and the ability to redeploy it if adaptations or upgrades are required, and data gathering. In other words, adaptations, upgrades and deployment of monitoring systems, in ASTREA, can occur as many times as required, independently of the number of nodes involved.

Microservices dynamic deployment at runtime can not be achieved in a simulation environment (e.g., using ns-3 simulator). Therefore, prior to the evaluation, ASTREA implementation is deployed in a test environment to demonstrate its feasibility. A heart rate monitoring system (the case specification), with two actions (average and maximum with time windows of 15 minutes), has been designed with the ASTREACE tool. Subsequently, the monitoring system has been deployed by using the *deployment, and propagation of adaptations and upgrades* mechanism. The network infrastructure consists of one source node without enough computational capabilities to perform in-network operations, an intermediary node, a destination node, and two external nodes in which the case subversion service and the system service repository are deployed. Each node receives the specification and microservices it requires to compose and get the monitoring system up and running. Data are collected by the sensor Polar H7 and gathered to the destination node through the *data gathering* mechanism. The components of each node are also shown, as well the microservices for getting the average and maximum values from the measurements, and the standardised interfaces that allow composition without recompilation. In relation to the implementation and in accordance with the framework conceptualisation, some spots in ASTREA have been identified as hot, and others as frozen.

The evaluation is focused on data transmission and propagation time in a highly dynamic network under severe conditions. To this end, the ns-3 simulator has been used, and in order to prevent possible influences of any random factors, different network configurations in terms of number of nodes of each role, volume of data generated, and three different mobility models (Random Walk mobility, RPGM, and Manhattan Grid mobility models) introduced

with BonnMotion technologies have been considered. Two studies have been designed to test (1) the time required for data that would correspond to a case specification and associated microservices to be transmitted (in a simulated form) and reach all network nodes involved in the monitoring system. This is carried out by *deployment, and propagation of adaptations and upgrades* mechanism; (2) success rate percentage of data gathered which is performed thought *data gathering* mechanism; and (3) energy cost of data transmission. In-network preprocessing actions have been introduced within the evaluation to compare how data gathering is affected by minimising the volume of data to be transmitted as long as the resulting information is useful information, and equal to that which would be obtained as a result of preprocessing at the destination nodes, but with the disadvantage of having to transmit the total raw data thought the networks. In other words, the transmission of raw data has been compared to the transmission of useful, already processed information. In addition, the success rates of data gathering with three different priorities and without priority have been contrasted, and the results show that in extreme circumstances where it is necessary to prioritise which data is retained, the data with the highest priority will prevail.

Finally, based on the evaluation results, it can be noted that in an area/arena of 4800m^2 a monitoring system could be adapted and upgraded in less than 18 seconds when there are 5 or more intermediary nodes in network infrastructure. It is also confirmed that the prioritisation of data is important, as the percentage of data gathered decreases as the priority of the data decreases. Concerning the relevance of performing in-network preprocessing, when it is not applied, an average success rate of 67.49% is achieved, and when it is applied, 80.74% is reached. Mobility models affect data transmission, and finally, the incorporation of an intermediary node or destination into the network has a different impact.

7.2 Future Work

We plan to develop the following points as possible lines of future work for this thesis:

- According to the proposals reviewed in Chapter 3, an improvement in data gathering with MEs can be achieved by applying sensor clustering. In this way, a ME does not have to visit each sensor to collect its data, but only the cluster heads, thus reducing the number of nodes to be visited.

- Also for data gathering, directed MEs could be included, whose unique purpose would be to gather data. Therefore, ASTREA could be extended with a set of services which would support the computation of routes for these MEs. This is an interesting line of

research, given that hardware improvements in these types of devices are increasing (better autonomy, control and reliability).

- The ASTREA framework could be applied in other application domains, such as large areas of crop fields, agriculture, livestock and renewable energy fields, among others. In these scenarios, the inclusion of other types of MEs, such as aerial drones, could be of interest. However, aerial drones present differences in mobility models because the movements do not have to be in the same plane, but can be three-dimensional. Their speed could also be higher than that of the mobility models used in this thesis.

- In terms of evaluation, the next step would be to deploy a monitoring application in a controlled environment. For this purpose, different testbeds can be explored. This test can be an option to work in a real, medium-sized deployment environment, while it could also provide feedback on the performance of the framework. In addition, other scenarios to those already evaluated can be implemented in the simulator. For example, it could be interesting to evaluate the performance in the same terms of those included in Chapter 6 when there is no multi-hop connection or when there is also a multi-hop connection between the source nodes. Moreover, larger scale scenarios both in terms of area and number of nodes could also be evaluated. It should be noted that this would not affect the implementation of the framework either monitoring systems, it would only be a modification at network infrastructure level that could affect the performance results.

- As for the framework and its extension, additional preprocessing and measurement microservices can be included, adding new sensors in ASTREA and making it available on various platforms. For this purpose, a methodology should be designed and made available to third party developers, allowing them to extend the framework independently.

Chapter 8

Conclusiones y Trabajo Futuro

8.1 Conclusiones

En este trabajo de tesis, se propone el *framework* ASTREA como una solución integral para desarrollar sistemas de monitorización adaptables y extensibles a partir de la composición y reutilización de software. Estos sistemas de monitorización deben soportar adaptaciones y actualizaciones posteriores al despliegue en tiempo de ejecución, así como la recopilación de datos. Además, tales sistemas tienen que ser funcionales y deben poder desplegarse en una infraestructura de red dinámica y móvil. Para lograr esto, se ha llevado a cabo un estudio de fundamentos, tecnologías, trabajos de investigación que abordan puntos relacionados con nuestro objetivo principal y las metas definidas. Además, también se han analizado cuestiones técnicas de desarrollo de software, dispositivos hardware y redes.

Durante el desarrollo de esta tesis, se han estudiado los dominios de aplicación de LES, fibromialgia y fetos PEG y bebés con bajo peso. Además, se ha considerado la monitorización del entorno como parte de la información de contexto. Se han desarrollado dos sistemas de monitorización para tales dominios, que en concreto se han aplicado en bebés PEG y en un estudio para evaluar la calidad del sueño de personas sufrían LES. Los requisitos de los sistemas se han definido con la colaboración de expertos (psicólogos, médicos, enfermeras, matronas, educadores y terapeutas).

Para monitorizar el contexto ambiental, hemos desarrollado un sistema de salud móvil (mSalud) que integra la aplicación CEnMO para tomar medidas de luminosidad, temperatura ambiental, ruido, humedad y presión desde un dispositivo móvil con sensores incorporados. Los datos recogidos con CEnMO y los datos fisiológicos recogidos con un actígrafo son información objetiva porque la recogida de estas mediciones no requiere de intervención humana. Esta información objetiva, se combinó con información recogida en cuestionarios

estandarizados (información subjetiva) para realizar el estudio de la calidad del sueño. En este caso, la monitorización ambiental ayudó a comprobar la validez ecológica de los datos y a estudiar cómo los factores ambientales afectan a la calidad del sueño.

Posteriormente, desarrollamos mPOP, otro sistema de mSalud para ofrecer un programa psicoeducativo a mujeres embarazadas, y a sus parejas, con fetos de bajo peso para la edad gestacional o bebés nacidos con bajo peso. Estos programas se ofrecen a través de tres aplicaciones móviles (VIVEmbarazo, VIVECrianza Hospitalización y VIVECrianza No Hospitalización). Los programas psicoeducativos han sido desarrollados por expertos en la materia, y pueden personalizarse en función de la evolución de los fetos, los bebés y sus padres. La información de los padres también es gestionada por un experto en la materia a través de una herramienta de autoría. Todo ello se apoya en el módulo AdaptEn, que comprende principalmente el conjunto de servicios que se orquestan para cargar y permitir la fácil modificación de los programas, así como garantizar la seguridad de la información.

Los ámbitos de aplicación abordados han servido para identificar características comunes, similitudes, requisitos, soluciones de software y dispositivos de hardware. Todos ellos nos han ayudado a adquirir los conocimientos y la base para proponer y evaluar nuevas ideas, las cuales han sido el punto de partida para el núcleo de la propuesta, el *framework* ASTREA. ASTREA ha sido creado para ayudar a los desarrolladores de software de un dominio específico en el diseño y desarrollo de sistemas de monitorización que incluyan sensores y wearables para monitorizar información de contexto (variables fisiológicas y factores ambientales). ASTREA proporciona soluciones a una serie de problemas, requisitos y objetivos comunes de los sistemas de monitorización modernos, fomentando la reutilización de software y automatizando ciertas tareas que en la mayoría de los sistemas deben llevarse a cabo, tales como el despliegue, la adaptación y las actualizaciones de los sistemas, y la recopilación de datos.

Dentro de ASTREA, en la definición de los sistemas de monitorización se especifican como *casos* individuales que pueden ser diseñados por un diseñador que podría ser un desarrollador de software de dominio específico, pero quien no tiene que tener conocimientos técnicos o de programación. A partir del concepto de *caso*, es posible definir la funcionalidad de los sistemas de monitorización, los dispositivos (sensores y wearables) que se van a utilizar, establecer relaciones entre la funcionalidad y los dispositivos, y establecer ciertos parámetros de configuración. Estos son elementos que están disponibles en la herramienta de edición visual ASTREACE, desarrollada para facilitar el proceso de diseño. El diseñador de casos puede seleccionar un elemento, y arrastrarlo y soltarlo en un lienzo para configurarlo. El resultado final será un diagrama de flujo que define la especificación de *caso* en formato JSON.

Los sistemas de monitorización desarrollados con ASTREA pueden operar en entornos no móviles y no cambiantes, pero el reto es justo el contrario. Con ASTREA pretendemos aportar soluciones a la propagación de adaptaciones y actualizaciones en tiempo de ejecución para los sistemas de monitorización distribuidos, ya operativos, que se despliegan en redes dinámicas y móviles. Por tanto, el *servicio de subversión de casos* genera diferentes subversiones a partir de la especificación de *caso* y de acuerdo a las capacidades de cada dispositivo (o nodo) que puede formar parte de la infraestructura de red. En ASTREA, los nodos se categorizan en tres roles diferentes (fuentes, intermediarios y destinos). En concreto, los *fuentes* recogen datos; los *intermediarios*, normalmente con mayores capacidades que los nodos fuente, son portadores de datos y pueden realizar operaciones en la red; y los *destinos* son los nodos con mayores capacidades de la infraestructura, que actúan como puerta de entrada entre Internet o la Nube, siendo el punto de entrada para el despliegue del caso.

Una vez diseñado el sistema, hay que desplegarlo en los dispositivos, es decir, instalar los microservicios y establecer la configuración. Estas tareas se vuelven más tediosas cuanto mayor es el número de dispositivos implicados y el número de sistemas de monitorización que hay que desplegar. Además, las adaptaciones y actualizaciones son necesarias porque las necesidades de los usuarios pueden cambiar, o porque el sistema necesita ser ampliado, o incluso porque el contexto cambia. Los cambios de contexto pueden implicar una adaptación autónoma en el sistema de monitorización que se gestiona mediante planes definidos dentro de la especificación del *caso*. Además, ASTREA incluye el mecanismo de *despliegue, y propagación de adaptaciones y actualizaciones* que soporta el despliegue autónomo del sistema, por lo que los desarrolladores de software de dominio específico deben especificar lo que consideren que debe ser cambiado, y la especificación de casos se propaga a través de la infraestructura de red dinámica, junto con los microservicios asociados que compondrán el sistema de monitorización de forma autónoma y en tiempo de ejecución, sin necesidad de recompilar ninguno de sus componentes. Estos microservicios deben haber sido desarrollados previamente por un desarrollador de software, y se almacenan en un repositorio para realizar la movilidad del código. La propagación de las adaptaciones y actualizaciones comienza en los nodos de destino y termina en los nodos de origen. Aquí, utilizamos elementos móviles para propagar las adaptaciones y actualizaciones a los nodos de la red que no están directamente en el rango de los nodos de destino.

La recogida de datos sigue presentando problemas de rendimiento en cuanto a la tasa de éxito de la recogida de datos, el ancho de banda y el consumo de energía. ASTREA implementa el mecanismo *recopilación de datos* para realizar operaciones de preprocesamiento en la red con el fin de obtener información útil a partir de los datos sin procesar, y así minimizar el

volumen de datos a transmitir lo antes posible. También, aprovechamos los elementos móviles para que actúen como portadores de datos y realicen el preprocesamiento en red de los datos de los que disponen. Además, el mecanismo *recopilación de datos* incluye la priorización de datos, el almacenamiento en búfer de datos y la política de reenvío de datos, por lo que la adaptación aquí también está presente.

El sistema de monitorización (ASTREAMS) es el núcleo de los mecanismos desarrollados. ASTREAMS integra varios microservicios clave entre los que destaca un orquestador (el gestor de casos), y gestores de datos y preprocesamiento. Estos microservicios son responsables de la coordinación, la composición de casos, la gestión de datos y la realización de operaciones dentro de la red, si las capacidades del nodo lo permiten. También cabe destacar la política de reenvío de datos, que se encarga de elegir el nodo más adecuado de rol superior al que se le envían los datos, cuando hay varias alternativas.

Se ha descrito un escenario de motivación para mostrar el comportamiento de usuarios y cuidadores en una residencia de ancianos en la que ASTREA puede ser útil. Los usuarios disponen de BSN para recoger información fisiológica, también hay sensores inalámbricos en puntos específicos para recoger medidas de factores ambientales, y los cuidadores llevan dispositivos móviles que actúan como portadores de datos desde los nodos origen hasta los nodos donde se centraliza la información (los nodos de destino) con mayor capacidad de cálculo, situados en el borde de la red (la pasarela). Cada residente puede requerir un sistema de monitorización específico, que incluso puede variar significativamente en función de su estado de salud. El *framework* ASTREA abarca tanto el despliegue autónomo del caso (el sistema de monitorización) como la capacidad de volver a desplegarla si se requieren adaptaciones o actualizaciones, y la recopilación de datos. En otras palabras, las adaptaciones, actualizaciones y despliegue de los sistemas de monitorización, en ASTREA, puede ocurrir tantas veces como sea necesario, independientemente del número de nodos implicados.

El despliegue dinámico de los microservicios en tiempo de ejecución no puede realizarse en un entorno de simulación (por ejemplo, utilizando el simulador ns-3). Por lo tanto, antes de la evaluación, la implementación de ASTREA es desplegada en un entorno de prueba para demostrar su viabilidad. Se ha diseñado con la herramienta ASTREACE un sistema de monitorización de la frecuencia cardíaca (la especificación del caso), con dos acciones (media y máxima con ventanas de tiempo de 15 minutos). Posteriormente, el sistema de monitorización se ha desplegado con el mecanismo de *despliegue, y propagación de adaptaciones y actualizaciones*. La infraestructura de la red consiste en un nodo fuente sin capacidad de cálculo suficiente para realizar operaciones en la red, un nodo intermediario, un nodo destino y dos nodos externos en los cuales se despliega el servicio de subversión de casos y el servicio de repositorio del

sistema. Cada nodo recibe la especificación y los microservicios que necesita para componer y poner en marcha el sistema de monitorización. Los datos son recogidos por el sensor Polar H7 y reunidos en el nodo destino a través del mecanismo *recopilación de datos*. También se muestran los componentes de cada nodo, así como los microservicios para obtener la media y máximo de los datos recogidos, y las interfaces estandarizadas que permiten la composición sin recompilación. En relación con la implementación y de acuerdo con la conceptualización de , se han identificado algunos puntos de ASTREA como calientes, y otros como fríos.

La evaluación se centra en la transmisión de datos y el tiempo de propagación en una red altamente dinámica bajo condiciones severas. Para ello, se ha utilizado el simulador ns-3 y, para evitar la posible influencia de cualquier factor aleatorio, se han considerado diferentes configuraciones de red en cuanto a número de nodos de cada rol, volumen de datos generados y tres modelos de movilidad diferentes (los modelos de movilidad Random Walk, RPGM y Manhattan Grid) introducidos con tecnologías BonnMotion. Se han diseñado dos estudios para comprobar (1) el tiempo necesario para que se transmitan los datos que corresponderían a una especificación de caso y microservicios asociados (de forma simulada), y estos lleguen a todos los nodos de la red involucrados en el sistema de monitorización. Esto se realiza mediante el mecanismo de *despliegue, y propagación de adaptaciones y actualizaciones*; (2) el porcentaje de éxito de los datos recogidos que se realiza mediante el mecanismo de *recopilación de datos*; y (3) el coste energético de la transmisión de datos.

En la evaluación se han introducido acciones de preprocesamiento en la red para comparar cómo afecta a la recogida de datos el minimizar el volumen de datos a transmitir siempre que la información resultante sea información útil, e igual a la que se obtendría como resultado del preprocesamiento en los nodos destino, pero con el inconveniente de tener que transmitir el total de los datos en bruto a través de la red. En otras palabras, se ha comparado la transmisión de datos en bruto con la transmisión de información útil ya procesada. Además, se han contrastado los porcentajes de éxito en la recogida de datos con tres prioridades diferentes y sin prioridad, y los resultados muestran que en circunstancias extremas en las que sea necesario priorizar qué datos se conservan, prevalecerán los de mayor prioridad.

Por último, en base a los resultados de la evaluación, se puede observar que en un área/escenario de 4800m^2 un sistema de monitorización podría adaptarse y actualizarse en menos de 18 segundos cuando hay 5 o más nodos intermedios en la infraestructura de la red. También se confirma que la priorización de los datos es importante, ya que el porcentaje de datos recopilados disminuye a medida que disminuye la prioridad de los mismos. En cuanto a la relevancia de realizar un preprocesamiento en la red, cuando no se aplica se consigue un porcentaje medio de éxito del 67,49%, y cuando se aplica se alcanza el 80,74%. Los modelos de movilidad afectan

a la transmisión de datos, y por último, la incorporación en la red de un nodo intermediario o destino tienen un impacto diferente.

8.2 Trabajo Futuro

Tenemos previsto desarrollar los siguientes puntos como posibles líneas de trabajo futuro para esta tesis:

- De acuerdo con las propuestas revisadas en el Capítulo 3, se puede conseguir una mejora en la recogida de datos con los elementos móviles aplicando el clustering de sensores. De esta forma, un elementos móviles no tiene que visitar cada sensor para recoger sus datos, sino sólo los líderes del cluster, reduciendo así el número de nodos a visitar.

- Además, para la recogida de datos, podrían incluirse elementos móviles dirigidos, cuyo único propósito sería recoger datos. Por lo tanto, ASTREA podría ampliarse con un conjunto de servicios que apoyen el cálculo de rutas para estos elementos móviles. Esta es una línea de investigación interesante, dado que las mejoras de hardware en este tipo de dispositivos son cada vez mayores (mayor autonomía, control y fiabilidad).

- El *framework* ASTREA podría aplicarse en otros dominios de aplicación, tales como grandes áreas de campos de cultivo, agricultura, ganadería y campos de energía renovable, entre otros. En estos escenarios, la inclusión de otros tipos de elementos móviles, como drones aéreos, podría ser de interés. Sin embargo, los drones aéreos presentan diferencias en los modelos de movilidad debido a que los movimientos no tienen que ser en el mismo plano, sino que pueden ser tridimensionales. Su velocidad también podría ser mayor que la de los modelos de movilidad utilizados en esta tesis.

- En términos de evaluación, el siguiente paso sería desplegar una aplicación de monitorización en un entorno controlado. Para ello, se pueden explorar diferentes bancos de pruebas. Esta prueba puede ser una opción para trabajar en un entorno real de despliegue de tamaño medio, a la vez que podría proporcionar información sobre el rendimiento del *framework*. Además, se pueden implementar en el simulador otros escenarios a los ya evaluados. Por ejemplo, podría ser interesante evaluar el rendimiento en los mismos términos incluídos en el capítulo 6 cuando no hay conexión multisalto o cuando también hay una conexión multisalto entre los nodos de origen. Además, podrían evaluarse escenarios de mayor escala, tanto en términos de área como de número de nodos. Cabe señalar que esto no afectaría a la implementación del *framework* ni a los sistemas de

monitorización, sólo sería una modificación a nivel de infraestructura de red que podría afectar a los resultados de rendimiento.

- En cuanto al *framework* y su extensión, se pueden incluir microservicios adicionales de preprocesamiento y medición, añadiendo nuevos sensores en ASTREA y habilitándolo en varias plataformas. Para ello, debería diseñarse una metodología y ponerla a disposición de los desarrolladores de terceros, permitiéndoles ampliar el *framework* de forma independiente.

References

[1] Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, 10 (7):1497 – 1516, 2012. ISSN 1570-8705.

[2] Abend, G. The meaning of 'theory'. *Sociological theory*, 26(2):173–199, 2008.

[3] Ahern, D. K., Kreslake, J. M., and Phalen, J. M. What is ehealth (6): perspectives on the evolution of ehealth research. *Journal of medical Internet research*, 8(1):e4, 2006.

[4] Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. Wireless sensor networks: a survey. *Computer networks*, 38(4):393–422, 2002.

[5] Alavi, A. H., Jiao, P., Buttlar, W. G., and Lajnef, N. Internet of things-enabled smart cities: State-of-the-art and future trends. *Measurement*, 129:589–606, 2018.

[6] Alderdice, F., Lynn, F., and Lobel, M. A review and psychometric evaluation of pregnancy-specific stress measures. *Journal of Psychosomatic Obstetrics & Gynecology*, 33(2):62–77, 2012.

[7] Alférez, G. H., Pelechano, V., Mazo, R., Salinesi, C., and Diaz, D. Dynamic adaptation of service compositions with variability models. *Journal of Systems and Software*, 91: 24–47, 2014.

[8] Alnuaimi, M., Shuaib, K., Alnuaimi, K., and Abdel-Hafez, M. Ferry-based data gathering in wireless sensor networks with path selection. *Procedia Computer Science*, 52:286–293, 2015.

[9] Altoaimy, L., Kurdi, H., Alromih, A., Alomari, A., Alrogi, E., and Ahmed, S. H. Enhanced distance-based gossip protocols for wireless sensor networks. In *2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–4. IEEE, 2019.

[10] Amoui, M., Derakhshanmanesh, M., Ebert, J., and Tahvildari, L. Achieving dynamic adaptation via management and interpretation of runtime models. *Journal of Systems and Software*, 85(12):2720–2737, 2012.

[11] Apple. Sector sanitario - apple watch - apple (es). URL https://www.apple.com/es/healthcare/apple-watch/.

[12] Argent, R. M., Voinov, A., Maxwell, T., Cuddy, S. M., Rahman, J. M., Seaton, S., Vertessy, R., and Braddock, R. D. Comparing modelling frameworks–a workshop approach. *Environmental Modelling & Software*, 21(7):895–910, 2006.

[13] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., et al. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.

[14] Arsanjani, A. Service-oriented modeling and architecture. *IBM developer works*, 1:15, 2004.

[15] Aslam, M. S., Rea, S., and Pesch, D. Service provisioning for the WSN cloud. In *2012 IEEE Fifth International Conference on Cloud Computing*, pages 962–969. IEEE, 2012.

[16] Avrora. The AVR simulation and analysis framework. URL http://compilers.cs.ucla.edu/avrora/.

[17] Bajaj, L., Takai, M., Ahuja, R., Tang, K., Bagrodia, R., and Gerla, M. Glomosim: A scalable network simulation environment. *UCLA computer science department technical report*, 990027(1999):213, 1999.

[18] Bąk, K., Diskin, Z., Antkiewicz, M., Czarnecki, K., and Wąsowski, A. Clafer: unifying class and feature modeling. *Software & Systems Modeling*, 15(3):811–845, 2016.

[19] Bakken, D. Middleware. *Encyclopedia of Distributed Computing*, 11, 2001.

[20] Balderas-Díaz, S., Benghazi, K., Garrido, J. L., Guerrero-Contreras, G., and Miró, E. Designing new low-cost home-oriented systems for monitoring and diagnosis of patients with sleep apnea-hypopnea. In *ICTs for Improving Patients Rehabilitation Research Techniques*, pages 210–221. Springer, 2014.

[21] Balderas-Díaz, S., Benghazi, K., Garrido, J. L., O'Hare, G. M., and Guerrero-Contreras, G. Integrating a dual method on a general architecture to self-adaptive monitoring systems. In *World Conference on Information Systems and Technologies*, pages 528–538. Springer, 2017.

[22] Balderas-Díaz, S., Martínez, M. P., Guerrero-Contreras, G., Miró, E., Benghazi, K., Sánchez, A. I., Garrido, J. L., and Prados, G. Using actigraphy and mHealth systems for an objective analysis of sleep quality on systemic lupus erythematosus patients. *Methods of information in medicine*, 56(02):171–179, 2017.

[23] Balderas-Díaz, S., Rodríguez-Fórtiz, M. J., Garrido, J. L., Bellido-González, M., and Guerrero-Contreras, G. Design of an adaptable mHealth system supporting a psycho-educational program for pregnant women with SGA foetuses. In *International Conference on Conceptual Modeling*, pages 125–135. Springer, 2021.

[24] Baloch, Z., Shaikh, F. K., and Unar, M. A. A context-aware data fusion approach for health-IoT. *International Journal of Information Technology*, 10(3):241–245, 2018.

[25] Bandyopadhyay, S., Sengupta, M., Maiti, S., and Dutta, S. Role of middleware for internet of things: A study. *International Journal of Computer Science and Engineering Survey*, 2(3):94–105, 2011.

[26] Bandyopadhyay, S., Sengupta, M., Maiti, S., and Dutta, S. A survey of middleware for internet of things. In *Recent trends in wireless and mobile networks*, pages 288–296. Springer, 2011.

[27] Barricelli, B. R., Cassano, F., Fogli, D., and Piccinno, A. End-user development, end-user programming and end-user software engineering: A systematic mapping study. *Journal of Systems and Software*, 149:101–137, 2019.

[28] Basford, L. and Slevin, O. *Theory and practice of nursing: An integrated approach to caring practice*. Nelson Thornes, 2003.

[29] Bass, L., Clements, P., and Kazman, R. *Software architecture in practice*. Addison-Wesley Professional, 2003.

[30] Bass, L., Weber, I., and Zhu, L. *DevOps: A software architect's perspective*. Addison-Wesley Professional, 2015.

[31] Bellido-González, M., Díaz-López, M. Á., López-Criado, S., and Maldonado-Lozano, J. Cognitive functioning and academic achievement in children aged 6–8 years, born at term after intrauterine growth restriction and fetal cerebral redistribution. *Journal of pediatric psychology*, 42(3):345–354, 2017.

[32] Bellido-González, M., Robles-Ortega, H., Castelar-Ríos, M. J., Díaz-López, M. Á., Gallo-Vallejo, J. L., Moreno-Galdó, M. F., and de Los Santos-Roig, M. Psychological distress and resilience of mothers and fathers with respect to the neurobehavioral performance of small-for-gestational-age newborns. *Health and quality of life outcomes*, 17 (1):1–13, 2019.

[33] Biggerstaff, T. and Richter, C. Reusability framework, assessment, and directions. *IEEE software*, 4(2):41, 1987.

[34] Birman, K. The promise, and limitations, of gossip protocols. *ACM SIGOPS Operating Systems Review*, 41(5):8–13, 2007.

[35] Bishop, T. A. and Karne, R. K. A survey of middleware. In *Computers and Their Applications*, pages 254–258, 2003.

[36] Blair, G. S., Coulson, G., Andersen, A., Blair, L., Clarke, M., Costa, F., Duran-Limon, H., Fitzpatrick, T., Johnston, L., Moreira, R., et al. The design and implementation of open ORB 2. *IEEE Distributed Systems Online*, 2(6):1–40, 2001.

[37] Bonato, P. Wearable sensors and systems. *IEEE Engineering in Medicine and Biology Magazine*, 29(3):25–36, 2010.

[38] Bonfitto, S., Hachem, F., Belay, E. G., Valtolina, S., and Mesiti, M.

[39] Bonomi, F., Milito, R., Zhu, J., and Addepalli, S. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16, 2012.

[40] BoonMotion. A mobility scenario generation and analysis tool. URL http://sys.cs.uos.de/bonnmotion/.

[41] Borgia, E. The internet of things vision: Key features, applications and open issues. *Computer Communications*, 54:1 – 31, 2014. ISSN 0140-3664.

[42] Bratman, M. E., Israel, D. J., and Pollack, M. E. Plans and resource-bounded practical reasoning. *Computational intelligence*, 4(3):349–355, 1988.

[43] Bronfenbrenner, U. Toward an experimental ecology of human development. *American psychologist*, 32(7):513, 1977.

[44] Brun, Y., Serugendo, G. D. M., Gacek, C., Giese, H., Kienle, H., Litoiu, M., Müller, H., Pezzè, M., and Shaw, M. Engineering self-adaptive systems through feedback loops. In *Software engineering for self-adaptive systems*, pages 48–70. Springer, 2009.

[45] Bukhsh, Z. A., van Sinderen, M., and Singh, P. M. SOA and EDA: A comparative study: Similarities, differences and conceptual guidelines on their usage. In *2015 12th International Joint Conference on e-Business and Telecommunications (ICETE)*, volume 2, pages 213–220. IEEE, 2015.

[46] Capilla, R., Bosch, J., Trinidad, P., Ruiz-Cortés, A., and Hinchey, M. An overview of dynamic software product line architectures and techniques: Observations from research and industry. *Journal of Systems and Software*, 91:3–23, 2014.

[47] Casadei, R., Fortino, G., Pianini, D., Russo, W., Savaglio, C., and Viroli, M. Modelling and simulation of opportunistic IoT services with aggregate computing. *Future Generation Computer Systems*, 91:252–262, 2019.

[48] Castro, L. A., Favela, J., Quintana, E., and Perez, M. Behavioral data gathering for assessing functional status and health in older adults using mobile phones. *Personal and Ubiquitous Computing*, 19(2):379–391, 2015.

[49] Cedillo, P., Sanchez, C., Campos, K., and Bermeo, A. A systematic literature review on devices and systems for ambient assisted living: solutions and trends from different user perspectives. In *2018 International Conference on eDemocracy & eGovernment (ICEDEG)*, pages 59–66. IEEE, 2018.

[50] Cerny, T. Aspect-oriented challenges in system integration with microservices, SOA and IoT. *Enterprise Information Systems*, 13(4):467–489, 2019.

[51] Cerny, T., Donahoo, M. J., and Trnka, M. Contextual understanding of microservice architecture: current and future directions. *ACM SIGAPP Applied Computing Review*, 17(4):29–45, 2018.

[52] Chahal, R. K., Kumar, N., and Batra, S. Trust management in social internet of things: A taxonomy, open issues, and challenges. *Computer Communications*, 150:13–46, 2020.

[53] Chakraborty, S. and Agarwal, Y. K. Solving the team orienteering problem with non-identical agents: A Lagrangian approach. *Networks*, 2021.

[54] Chandrasekhara, P. K. S., Jayachandran, N. V., Rajasekhar, L., Thomas, J., and Narsimulu, G. The prevalence and associations of sleep disturbances in patients with systemic lupus erythematosus. *Modern rheumatology*, 19(4):407–415, 2009.

[55] Chen, L., Zhou, P., Gao, L., and Xu, J. Adaptive fog configuration for the industrial Internet of Things. *IEEE Transactions on Industrial Informatics*, 14(10):4656–4664, 2018.

[56] Chesson Jr, A. L., Ferber, R. A., Fry, J. M., Grigg-Damberger, M., Hartse, K. M., Hurwitz, T. D., Johnson, S., Kader, G. A., Littner, M., Rosen, G., et al. The indications for polysomnography and related procedures. *Sleep*, 20(6):423–487, 1997.

[57] Chopra, N., Kalia, A., and Thakur, J. Performance analysis of network simulation tools: NS-2 and J-SIM. *Journal of Open Source Developments*, 2(3):5–11, 2015.

[58] Cook, D. J., Augusto, J. C., and Jakkula, V. R. Ambient intelligence: Technologies, applications, and opportunities. *Pervasive and Mobile Computing*, 5(4):277–298, 2009.

[59] Corral, J. M. R., Ruiz-Rube, I., Balcells, A. C., Mota-Macías, J. M., Morgado-Estévez, A., and Dodero, J. M. A study on the suitability of visual languages for non-expert robot programmers. *IEEE access*, 7:17535–17550, 2019.

[60] Crespi, V., Galstyan, A., and Lerman, K. Top-down vs bottom-up methodologies in multi-agent system design. *Autonomous Robots*, 24(3):303–313, 2008.

[61] CubeSensors. CubeSensors - feel better. URL https://cubesensors.com/.

[62] da Costa Vieira Rezende, J., da Silva, R. I., and Souza, M. J. F.

[63] De Gelder, B. and Bertelson, P. Multisensory integration, perception and ecological validity. *Trends in cognitive sciences*, 7(10):460–467, 2003.

[64] De Lemos, R., Giese, H., Müller, H. A., Shaw, M., Andersson, J., Litoiu, M., Schmerl, B., Tamura, G., Villegas, N. M., Vogel, T., et al. Software engineering for self-adaptive systems: A second research roadmap. In *Software Engineering for Self-Adaptive Systems II*, pages 1–32. Springer, 2013.

[65] Dey, A. K. Understanding and using context. *Personal and ubiquitous computing*, 5(1): 4–7, 2001.

[66] Dey, A. K., Salber, D., Abowd, G. D., and Futakawa, M. The conference assistant: Combining context-awareness with wearable computing. In *Digest of Papers. Third International Symposium on Wearable Computers*, pages 21–28. IEEE, 1999.

[67] Diaz, S. and Mendez, D. Dynamic minimum spanning tree construction and maintenance for wireless sensor networks. *Revista Facultad de Ingeniería Universidad de Antioquia*, (93):57–69, 2019.

[68] Dictionary, O. A. L. Middleware noun - definition, pictures, pronunciation and usage notes | oxford advanced learner's dictionary at oxfordlearnersdictionaries.com. URL https://www.oxfordlearnersdictionaries.com/definition/english/middleware.

[69] Dooley, L. M. Case study research and theory building. *Advances in developing human resources*, 4(3):335–354, 2002.

[70] Dragone, M., Amato, G., Bacciu, D., Chessa, S., Coleman, S., Di Rocco, M., Gallicchio, C., Gennaro, C., Lozano, H., Maguire, L., et al. A cognitive robotic ecology approach to self-configuring and evolving aal systems. *Engineering Applications of Artificial Intelligence*, 45:269–280, 2015.

[71] Ebert, C., Gallardo, G., Hernantes, J., and Serrano, N. Devops. *Ieee Software*, 33(3): 94–100, 2016.

[72] Elazhary, H. Internet of things (iot), mobile cloud, cloudlet, mobile iot, iot cloud, fog, mobile edge, and edge emerging computing paradigms: Disambiguation and research directions. *Journal of Network and Computer Applications*, 128:105–140, 2019.

[73] Empatica. E4 wristband | Real-time physiological signals | Wearable PPG, EDA, Temperature, Motion sensors, . URL https://www.empatica.com/research/e4.

[74] Empatica. Embrace2 Seizure Monitoring | Smarter Epilepsy Management | Embrace Watch | Empatica, . URL https://www.empatica.com/en-eu/embrace2/.

[75] Erl, T. *SOA principles of service design (the Prentice Hall service-oriented computing series from Thomas Erl)*. Prentice Hall PTR, 2007.

[76] Faheem, M., Butt, R. A., Raza, B., Ashraf, M. W., Ngadi, M. A., and Gungor, V. C.

[77] Fayad, M. and Schmidt, D. C. Object-oriented application frameworks. *Communications of the ACM*, 40(10):32–38, 1997.

[78] Fazio, M., Merlino, G., Bruneo, D., and Puliafito, A. An architecture for runtime customization of smart devices. In *2013 IEEE 12th International Symposium on Network Computing and Applications*, pages 157–164. IEEE, 2013.

[79] Fernando, N., Loke, S. W., and Rahayu, W. Mobile cloud computing: A survey. *Future generation computer systems*, 29(1):84–106, 2013.

[80] Finin, T., Fritzson, R., McKay, D., and McEntire, R. Kqml as an agent communication language. In *Proceedings of the third international conference on Information and knowledge management*, pages 456–463, 1994.

[81] Fonken, L. K. and Nelson, R. J. The effects of light at night on circadian clocks and metabolism. *Endocrine reviews*, 35(4):648–670, 2014.

[82] Fortino, G., Galzarano, S., Gravina, R., and Li, W.

[83] Fortino, G., Guerrieri, A., O'Hare, G. M., and Ruzzelli, A. A flexible building management framework based on wireless sensor and actuator networks. *Journal of Network and Computer Applications*, 35(6):1934–1952, 2012.

[84] Gallager, R. G., Humblet, P. A., and Spira, P. M. A distributed algorithm for minimum-weight spanning trees. *ACM Transactions on Programming Languages and systems (TOPLAS)*, 5(1):66–77, 1983.

[85] Galton, A. and Augusto, J. C. Two approaches to event definition. In *International Conference on Database and Expert Systems Applications*, pages 547–556. Springer, 2002.

[86] García-Hernández, C. F., Ibarguengoytia-Gonzalez, P. H., García-Hernández, J., and Pérez-Díaz, J. A. Wireless sensor networks and applications: a survey. *IJCSNS International Journal of Computer Science and Network Security*, 7(3):264–273, 2007.

[87] Garlan, D. Software architecture: a roadmap. In *Proceedings of the Conference on the Future of Software Engineering*, pages 91–101, 2000.

[88] Garlan, D., Cheng, S.-W., Huang, A.-C., Schmerl, B., and Steenkiste, P. Rainbow: Architecture-based self-adaptation with reusable infrastructure. *Computer*, 37(10): 46–54, 2004.

[89] Gimeno, M. Informe anual sobre el desarrollo de la sociedad de la información en españa. *Fundación Orange. Madrid: Fundación Orange*, 2014.

[90] Global, I. What is Methodological Framework | IGI Global. URL https://www.igi-global.com/dictionary/methodological-framework/18485.

[91] González, I. F., Maeso-Méndez, S., Miranda, A. S., del Hoyo Moracho, M., Blázquez, I. L., and López, I. D. Diferencias en la función tiroidea de los pequeños para la edad gestacional y los de peso adecuado.¿ Es normal la función tiroidea de los recién nacidos pequeños para la edad gestacional? In *Anales de Pediatría*. Elsevier, 2020.

[92] Gould, G., Whyte, K., Rhind, G., Airlie, M., Catterall, J., Shapiro, C., and Douglas, N. The sleep hypopnea syndrome. *American Review of Respiratory Disease*, 2012.

[93] Gravina, R., Alinia, P., Ghasemzadeh, H., and Fortino, G. Multi-sensor fusion in body sensor networks: State-of-the-art and research challenges. *Information Fusion*, 35:68 – 80, 2017.

[94] Greenfield, J. and Short, K. Software factories: assembling applications with patterns, models, frameworks and tools. In *Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, pages 16–27, 2003.

[95] Group, O. M. About the common object request broker architecture specification version 3.4. URL https://www.omg.org/spec/CORBA/3.4/About-CORBA/.

[96] Guerrero-Contreras, G., Garrido, J. L., Balderas-Diaz, S., and Rodríguez-Domínguez, C. A context-aware architecture supporting service availability in mobile cloud computing. *IEEE Transactions on Services Computing*, 10(6):956–968, 2017.

[97] Guerrero-Contreras, G., Garrido, J. L., Fórtiz, M. J. R., O'Hare, G. M. P., and Balderas-Díaz, S. Impact of Transmission Communication Protocol on a Self-adaptive Architecture for Dynamic Network Environments. In *Recent Advances in Information Systems and Technologies*, pages 115–124. Springer International Publishing, 2017.

[98] Guinard, D., Trifa, V., and Wilde, E. A resource oriented architecture for the web of things. In *2010 Internet of Things (IOT)*, pages 1–8. IEEE, 2010.

[99] Hakala, I. and Tan, X. A Statecharts-Based Approach for WSN Application Development. *Journal of Sensor and Actuator Networks*, 9(4):45, 2020.

[100] Haller, S., Karnouskos, S., and Schroth, C. The internet of things in an enterprise context. In *Future internet symposium*, pages 14–28. Springer, 2008.

[101] Handy, M., Haase, M., and Timmermann, D. Low energy adaptive clustering hierarchy with deterministic cluster-head selection. In *4th international workshop on mobile and wireless communications network*, pages 368–372. IEEE, 2002.

[102] Hawalah, A. and Fasli, M. Dynamic user profiles for web personalisation. *Expert Systems with Applications*, 42(5):2547–2569, 2015.

[103] Hayes-Roth, F. Architecture-based acquisition and development of software: Guidelines and recommendations from the arpa domain-specific software architecture (dssa) program. *Teknowledge Federal Systems. Version*, 1, 1994.

[104] He, H., Miao, H., Liang, Z., Zhang, Y., Jiang, W., Deng, Z., Tang, J., Liu, G., and Luo, X. Prevalence of small for gestational age infants in 21 cities in china, 2014–2019. *Scientific reports*, 11(1):1–10, 2021.

[105] He, X., Liu, S., Yang, G., and Xiong, N. Achieving efficient data collection in heterogeneous sensing wsns. *IEEE Access*, 6:63187–63199, 2018.

[106] Health, S. C. Small for gestational age. URL https://www.stanfordchildrens.org/es/topic/default?id=smallforgestationalage-90-P05520.

[107] Hill, J. W. and Powell, P. The national healthcare crisis: Is ehealth a key solution? *Business Horizons*, 52(3):265–277, 2009.

[108] Hinchey, M. G. and Sterritt, R. Self-managing software. *Computer*, 39(2):107–109, 2006.

[109] Hong, X., Gerla, M., Pei, G., and Chiang, C.-C. A group mobility model for ad hoc wireless networks. In *Proceedings of the 2nd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems - MSWiM '99*, pages 53–60. ACM Press, 1999.

[110] Huang, D. et al. Mobile cloud computing. *IEEE COMSOC Multimedia Communications Technical Committee (MMTC) E-Letter*, 6(10):27–31, 2011.

[111] Hub, I. C. L. What is SOA (Service-Oriented Architecture)? - India | IBM. URL https://www.ibm.com/in-en/cloud/learn/soa.

[112] Iaboni, A., Ibanez, D., Gladman, D. D., Urowitz, M. B., and Moldofsky, H. Fatigue in systemic lupus erythematosus: contributions of disordered sleep, sleepiness, and depression. *The Journal of rheumatology*, 33(12):2453–2457, 2006.

[113] Ibrahim, N. and Mouel, F. L. A survey on service composition middleware in pervasive environments. *arXiv preprint arXiv:0909.2183*, 2009.

[114] Iftikhar, M. U. and Weyns, D. ActivFORMS: A runtime environment for architecture-based adaptation with guarantees. In *2017 IEEE International Conference on Software Architecture Workshops (ICSAW)*, pages 278–281. IEEE, 2017.

[115] Institute, T. M. From iCalm to q sensor to physiio to empatica. URL https://affect.media.mit.edu/projectpages/iCalm/iCalm-2-Q.html.

[116] Instruments, T. CC2541SENSORTAG-RD reference design. URL https://www.ti.com/tool/CC2541SENSORTAG-RD.

[117] J-sim. J-sim. URL https://www.j-sim.org/.

[118] Jamsa, K. *Cloud computing: SaaS, PaaS, IaaS, virtualization, business models, mobile, security and more*. Jones & Bartlett Publishers, 2012.

[119] Jeong, Y., Joo, H., Hong, G., Shin, D., and Lee, S. AVIoT: Web-based interactive authoring and visualization of indoor internet of things. *IEEE Transactions on Consumer Electronics*, 61(3):295–301, 2015.

[120] Johnson, R. E. Documenting frameworks using patterns. In *conference proceedings on Object-oriented programming systems, languages, and applications*, pages 63–76, 1992.

[121] Johnson, R. E. Frameworks= (components+patterns). *Communications of the ACM*, 40 (10):39–42, 1997.

[122] Johnson, R. E. and Foote, B. Designing reusable classes. *Journal of object-oriented programming*, 1(2):22–35, 1988.

[123] Joorabchi, M. E., Mesbah, A., and Kruchten, P. Real challenges in mobile app development. In *2013 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 15–24. IEEE, 2013.

[124] Jovanov, E., Price, J., Raskovic, D., Kavi, K., Martin, T., and Adhami, R. Wireless personal area networks in telemedical environment. In *Proceedings 2000 IEEE EMBS International Conference on Information Technology Applications in Biomedicine. ITAB-ITIS 2000. Joint Meeting Third IEEE EMBS International Conference on Information Technol*, pages 22–27. IEEE, 2000.

[125] Kakousis, K., Paspallis, N., and Papadopoulos, G. A. A survey of software adaptation in mobile and ubiquitous computing. *Enterprise Information Systems*, 4(4):355–389, 2010.

[126] Kaluža, M., Troskot, K., and Vukelić, B. Comparison of front-end frameworks for web applications development. *Zbornik Veleučilišta u Rijeci*, 6(1):261–282, 2018.

[127] Kephart, J. O. and Chess, D. M. The vision of autonomic computing. *Computer*, 36(1): 41–50, 2003.

[128] Ketfi, A., Belkhatir, N., and Cunin, P.-Y. Automatic adaptation of component-based software. In *Second Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications*, pages 1365–1371. Citeseer, 2002.

[129] Khalid, Z., Khalid, U., Sarijari, M. A., Safdar, H., Ullah, R., Qureshi, M., and Rehman, S. U.

[130] Konstantopoulos, C., Vathis, N., Pantziou, G., and Gavalas, D.

[131] Konstantopoulos, C., Vathis, N., Pantziou, G., and Gavalas, D. Employing mobile elements for delay-constrained data gathering in WSNs. *Computer Networks*, 135: 108–131, 2018.

[132] Krafzig, D., Banke, K., and Slama, D. *Enterprise SOA: service-oriented architecture best practices*. Prentice Hall Professional, 2005.

[133] Kruchten, P. B. The 4+ 1 view model of architecture. *IEEE software*, 12(6):42–50, 1995.

[134] Krueger, C. W. Software reuse. *ACM Computing Surveys (CSUR)*, 24(2):131–183, 1992.

[135] Krupitzer, C., Roth, F. M., VanSyckel, S., Schiele, G., and Becker, C. A survey on engineering approaches for self-adaptive systems. *Pervasive and Mobile Computing*, 17: 184–206, 2015.

[136] Kulkarni, G., Shelke, R., Palwe, R., Solanke, V., Belsare, S., and Mohite, S. Mobile cloud computing-bring your own device. In *2014 Fourth International Conference on Communication Systems and Network Technologies*, pages 565–568. IEEE, 2014.

[137] Kumar, N. and Dash, D. Flow based efficient data gathering in wireless sensor network using path-constrained mobile sink. *Journal of Ambient Intelligence and Humanized Computing*, 11(3):1163–1175, 2020.

[138] Latré, B., Braem, B., Moerman, I., Blondia, C., and Demeester, P. A survey on wireless body area networks. *Wireless networks*, 17(1):1–18, 2011.

[139] Lima, J. C. D., da Rocha, C. C., Augustin, I., and Dantas, M. A. R. CARS-AD Project: Context-aware recommender system for authentication decision in pervasive and mobile environments. *Advances and Applications in Mobile Computing*, page 201, 2012.

[140] Lin, J., Yu, W., Zhang, N., Yang, X., Zhang, H., and Zhao, W. A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications. *IEEE Internet of Things Journal*, 4(5):1125–1142, 2017.

[141] Luckham, D. C. and Vera, J. An event-based architecture definition language. *IEEE transactions on Software Engineering*, 21(9):717–734, 1995.

[142] Luo, Y., Li, W., and Qiu, S. Anomaly detection based latency-aware energy consumption optimization for IoT data-flow services. *Sensors*, 20(1):122, 2020.

[143] Maes, P. The agent network architecture (ANA). *Acm sigart bulletin*, 2(4):115–120, 1991.

[144] Mahmood, M. A., Seah, W. K., and Welch, I. Reliability in wireless sensor networks: A survey and challenges ahead. *Computer Networks*, 79:166–187, 2015.

[145] Maker, F., Amirtharajah, R., and Akella, V. Meloades: Methodology for long-term online adaptation of embedded software for heterogeneous devices. *Journal of Systems Architecture*, 59(8):643–655, 2013.

[146] Malik, S. and Kim, D.-H. A comparison of RESTful vs. SOAP web services in actuator networks. In *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)*, pages 753–755. IEEE, 2017.

[147] Maréchaux, J.-L. Combining service-oriented architecture and event-driven architecture using an enterprise service bus. *IBM developer works*, 12691275, 2006.

[148] Maróti, M., Kusy, B., Simon, G., and Lédeczi, A. The flooding time synchronization protocol. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 39–49, 2004.

[149] Matos, S. N. and Fernandes, C. T. Early definition of frozen and hot spots in the development of domain frameworks. In *Fourteenth ACM SIGSOFT Symposium on Foundations of Software Engineering*, 2006.

[150] McIlroy, B. J. N. P. . R. B., M. D. Mass-produced software components. In *Proceedings of the 1st international conference on software engineering, Garmisch Pattenkirchen, Germany. 1968*, pages 88–98. Garmisch Pattenkirchen, 1968.

[151] McKinley, P. K., Sadjadi, S. M., Kasten, E. P., and Cheng, B. H. Composing adaptive software. *Computer*, 37(7):56–64, 2004.

[152] McMeekin, N., Wu, O., Germeni, E., and Briggs, A. How methodological frameworks are being developed: evidence from a scoping review. *BMC medical research methodology*, 20(1):1–9, 2020.

[153] Mehdi, M., Mühlmeier, G., Agrawal, K., Pryss, R., Reichert, M., and Hauck, F. J. Referenceable mobile crowdsensing architecture: A healthcare use case. *Procedia computer science*, 134:445–451, 2018.

[154] Mell, P., Grance, T., et al. The NIST definition of cloud computing. 2011.

[155] Michelson, B. M. Event-driven architecture overview. *Patricia Seybold Group*, 2(12): 10–1571, 2006.

[156] Microsoft. What can I still do with my Microsoft Band? URL https://support.microsoft.com/en-us/topic/what-can-i-still-do-with-my-microsoft-band-a2a59355-5be0-3441-9fff-4dc27bcbafb5.

[157] Miró, E., Martínez, M. P., Sánchez, A. I., Prados, G., and Medina, A. When is pain related to emotional distress and daily functioning in fibromyalgia syndrome? the mediating roles of self-efficacy and sleep quality. *British journal of health psychology*, 16(4):799–814, 2011.

[158] Miyazaki, T., Li, P., Guo, S., Kitamichi, J., Hayashi, T., and Tsukahara, T. On-demand customizable wireless sensor network. *Procedia Computer Science*, 52:302–309, 2015.

[159] Mottola, L., Picco, G. P., and Sheikh, A. A. Figaro: Fine-grained software reconfiguration for wireless sensor networks. In *European Conference on Wireless Sensor Networks*, pages 286–304. Springer, 2008.

[160] Muchandi, V. Applying 4+ 1 view architecture with uml 2. *FCGSS White Paper*, 2007.

[161] Munkittrick, K. and McCarty, L. An integrated approach to aquatic ecosystem health: top-down, bottom-up or middle-out? *Journal of aquatic ecosystem health*, 4(2):77–90, 1995.

[162] Murray, N. Technologies - Niall Murray. URL http://www.niallmurray.info/technologies.

[163] Muse. Muse 2 - Tu asistente personal de meditacion. URL https://choosemuse.com/es/muse-2/?store_id=eu&utm_source=Google&utm_medium=PaidSearch&utm_campaign=Shopping&gclid=Cj0KCQiA15yNBhDTARIsAGnwe0X7YNMH6NZGzoaaFq_MLidnAX3G5zt7RpKvthSft9VwiU7AbHYYuMoaAjDIEALw_wcB.

[164] Nadareishvili, I., Mitra, R., McLarty, M., and Amundsen, M. *Microservice architecture: aligning principles, practices, and culture*. " O'Reilly Media, Inc.", 2016.

[165] Naqvi, N. Z., kishore Ramakrishnan, A., Preuveneers, D., and Berbers, Y. Walking in the clouds: deployment and performance trade-offs of smart mobile applications for intelligent environments. In *2013 9th International Conference on Intelligent Environments*, pages 212–219. IEEE, 2013.

[166] Nast, I., Bolten, M., Meinlschmidt, G., and Hellhammer, D. H. How to measure prenatal stress? A systematic review of psychometric instruments to assess psychosocial stress during pregnancy. *Paediatric and perinatal epidemiology*, 27(4):313–322, 2013.

[167] Navarrete-Navarrete, N., Peralta-Ramírez, M., Sabio, J., Martínez-Egea, I., Santos-Ruiz, A., and Jiménez-Alonso, J. Quality-of-life predictor factors in patients with sle and their modification after cognitive behavioural therapy. *Lupus*, 19(14):1632–1639, 2010.

[168] Netsim. NetSim Network Simulator. URL https://www.boson.com/netsim-cisco-network-simulator.

[169] Neumitra. Neumitra. URL https://www.neumitra.com/.

[170] Newman, S. *Building microservices: designing fine-grained systems*. O'Reilly Media, Inc., 2015.

[171] Newsroom, S. What you may not know about galaxy s4 innovative technology – samsung global newsroom. URL https://news.samsung.com/global/what-you-may-not-know-about-galaxy-s4-innovative-techonology.

[172] Ng, K.-G., Ting, C.-M., Yeo, J.-H., Sim, K.-W., Peh, W.-L., Chua, N.-H., Chua, N.-K., and Kwong, F. Progress on the development of the mediwatch ambulatory blood pressure monitor and related devices. *Blood Pressure Monitoring*, 9(3):149–165, 2004.

[173] ns-3. ns-3 Network Simulator. URL https://www.nsnam.org/.

[174] OMNeT++. Omnet++ discrete event simulator. URL http://omnetpp.org/.

[175] OpNet. Optnet network simulator. URL https://opnetprojects.com/opnet-network-simulator/.

[176] Ortiz, Ó., García, A. B., Capilla, R., Bosch, J., and Hinchey, M. Runtime variability for dynamic reconfiguration in wireless sensor network product lines. In *Proceedings of the 16th International Software Product Line Conference-Volume 2*, pages 143–150, 2012.

[177] Oxipatch. Oxi tire patches. URL https://oxipatch.com/.

[178] O'Grady, M. J., O'Hare, G. M., and Donaghey, C. Delivering adaptivity through context-awareness. *Journal of Network and Computer Applications*, 30(3):1007–1033, 2007.

[179] O'Grady, M. J., Muldoon, C., Dragone, M., Tynan, R., and O'Hare, G. M. Towards evolutionary ambient assisted living systems. *Journal of Ambient Intelligence and Humanized Computing*, 1(1):15–29, 2010.

[180] Palagini, L., Tani, C., Mauri, M., Carli, L., Vagnani, S., Bombardieri, S., Gemignani, A., and Mosca, M. Sleep disorders and systemic lupus erythematosus. *Lupus*, 23(2): 115–123, 2014.

[181] Pallapa, G., Das, S. K., Di Francesco, M., and Aura, T. Adaptive and context-aware privacy preservation exploiting user interactions in smart environments. *Pervasive and Mobile Computing*, 12:232–243, 2014.

[182] Papazoglou, M. P. Service-oriented computing: Concepts, characteristics and directions. In *Proceedings of the Fourth International Conference on Web Information Systems Engineering, 2003. WISE 2003*, pages 3–12. IEEE, 2003.

[183] Pearce, M., Goel, A. K., Kolodner, I., Zimring, C., Sentosa, L., and Billington, R. Case-based design support: A case study in architectural design. *IEEE expert*, 7(5): 14–20, 1992.

[184] Peltz, C. Web services orchestration and choreography. *Computer*, 36(10):46–52, 2003.

[185] Perrey, R. and Lycett, M. Service-oriented architecture. In *2003 Symposium on Applications and the Internet Workshops, 2003. Proceedings*, pages 116–119. IEEE, 2003.

[186] Pfannemuller, M., Breitbach, M., Krupitzer, C., Weckesser, M., Becker, C., Schmerl, B., and Schurr, A.

[187] Polar. Polar H9 | Pulsómetro con banda pectoral económico | Polar España, . URL https://www.polar.com/es/productos/accesorios/pulsometro_H9.

[188] Polar. Polar H7 | user manual, . URL https://support.polar.com/e_manuals/H7_Heart_Rate_Sensor/Polar_H7_Heart_Rate_Sensor_accessory_manual_English__.pdf.

[189] Pons-Estel, G. J., Alarcón, G. S., Scofield, L., Reinlib, L., and Cooper, G. S. Understanding the epidemiology and progression of systemic lupus erythematosus. In *Seminars in arthritis and rheumatism*, volume 39, pages 257–268. Elsevier, 2010.

[190] Pramsohler, T., Schenk, S., Barthels, A., and Baumgarten, U. A layered interface-adaptation architecture for distributed component-based systems. *Future Generation Computer Systems*, 47:113–126, 2015.

[191] Radios, B. Blueradios provides low energy bluetooth 4.0 smart ready wireless sensors and single-mode modules for wireless medical patient monitoring. URL https://www.blueradios.com/hardware_sensors.htm.

[192] Rahmani, A. M., Gia, T. N., Negash, B., Anzanpour, A., Azimi, I., Jiang, M., and Liljeberg, P. Exploiting smart e-Health gateways at the edge of healthcare Internet-of-Things: A fog computing approach. *Future Generation Computer Systems*, 78:641–658, 2018.

[193] Rajput, A. and Brahimi, T. Characterizing internet of medical things/personal area networks landscape. In *Innovation in Health Informatics*, pages 353–371. Elsevier, 2020.

[194] Rhodes, B. J. The wearable remembrance agent: A system for augmented memory. *Personal Technologies*, 1(4):218–224, 1997.

[195] Rowland, S. P., Fitzgerald, J. E., Holme, T., Powell, J., and McGregor, A. What is the clinical value of mhealth for patients? *NPJ digital medicine*, 3(1):1–6, 2020.

[196] Rubenstein-Montano, B., Liebowitz, J., Buchwalter, J., McCaw, D., Newman, B., Rebeck, K., and Team, T. K. M. M. A systems thinking framework for knowledge management. *Decision support systems*, 31(1):5–16, 2001.

[197] Rudestam K. E., N. R. R. *Surviving your dissertation*. 1992.

[198] Ruparelia, N. B. Software development lifecycle models. *ACM SIGSOFT Software Engineering Notes*, 35(3):8–13, 2010.

[199] Ryu, S. Book review: mhealth: new horizons for health through mobile technologies: based on the findings of the second global survey on ehealth (global observatory for ehealth series, volume 3). *Healthcare Informatics Research*, 18(3):231–233, 2012.

[200] Saad, E., Awadalla, M., and Darwish, R. A data gathering algorithm for a mobile sink in large-scale sensor networks. In *2008 The Fourth International Conference on Wireless and Mobile Communications*, pages 207–213. IEEE, 2008.

[201] Sabatucci, L., Seidita, V., and Cossentino, M. The four types of self-adaptive systems: a metamodel. In *International Conference on Intelligent Interactive Multimedia Systems and Services*, pages 440–450. Springer, 2018.

[202] Salehie, M. and Tahvildari, L. Self-adaptive software: Landscape and research challenges. *ACM transactions on autonomous and adaptive systems (TAAS)*, 4(2):1–42, 2009.

[203] Samsung. Gear 2 Neo | Soporte Samsung CL. URL https://www.samsung.com/cl/support/model/SM-R3810ZOACHO/.

[204] Sanctis, M. D., Bucchiarone, A., and Marconi, A.

[205] Saravanan, K., Julie, E. G., and Robinson, Y. H. Smart cities & iot: evolution of applications, architectures & technologies, present scenarios & future dream. In *Internet of Things and Big Data Analytics for Smart Generation*, pages 135–151. Springer, 2019.

[206] Satyanarayanan, M. Fundamental challenges in mobile computing. In *Proceedings of the Fifteenth Annual ACM Symposium on Principles of Distributed Computing*, PODC '96, 1996.

[207] Satyanarayanan, M. Pervasive computing: Vision and challenges. *IEEE Personal communications*, 8(4):10–17, 2001.

[208] Schilit, B., Adams, N., and Want, R. Context-aware computing applications. In *1994 First Workshop on Mobile Computing Systems and Applications*, pages 85–90. IEEE, 1994.

[209] Schmuckler, M. A. What is ecological validity? A dimensional analysis. *Infancy*, 2(4): 419–436, 2001.

[210] Searle, J. R. and Searle, J. R. *Speech acts: An essay in the philosophy of language*, volume 626. Cambridge university press, 1969.

[211] Sedighiani, K., Shokrollahi, S., and Shams, F. Basba: a framework for building adaptable service-based applications. *Journal of Systems and Software*, 179:110989, 2021.

[212] Seneviratne, S., Hu, Y., Nguyen, T., Lan, G., Khalifa, S., Thilakarathna, K., Hassan, M., and Seneviratne, A. A survey of wearable devices and challenges. *IEEE Communications Surveys & Tutorials*, 19(4):2573–2620, 2017.

[213] Sensorcon. Buy portable co detectors & gas detectors | sensorcon – sensorcon - sensing solutions by molex. URL https://www.sensorcon.com/pages/co-collection.

[214] Shah, R., Roy, S., Jain, S., and Brunette, W. Data MULEs: modeling a three-tier architecture for sparse sensor networks. In *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications, 2003.*

[215] Shan, T. C. and Hua, W. W. Taxonomy of java web application frameworks. In *2006 IEEE International Conference on e-Business Engineering (ICEBE'06)*, pages 378–385. IEEE, 2006.

[216] Shaw, M. The coming-of-age of software architecture research. In *Proceedings of the 23rd international conference on Software engineering*, page 656. Citeseer, 2001.

[217] Shawish, A. and Salama, M. Cloud computing: paradigms and technologies. In *Inter-cooperative collective intelligence: Techniques and applications*, pages 39–67. Springer, 2014.

[218] Shevtsov, S., Berekmeri, M., Weyns, D., and Maggio, M. Control-theoretical software adaptation: A systematic literature review. *IEEE Transactions on Software Engineering*, 44(8):784–810, 2017.

[219] Shi, W., Cao, J., Zhang, Q., Li, Y., and Xu, L. Edge computing: Vision and challenges. *IEEE internet of things journal*, 3(5):637–646, 2016.

[220] Sinaeepourfard, A., Garcia, J., Masip-Bruin, X., Marín-Tordera, E., Cirera, J., Grau, G., and Casaus, F. Estimating smart city sensors data generation. In *2016 Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*, pages 1–8. IEEE, 2016.

[221] Sinclair, M. A guide to understanding theoretical and conceptual frameworks. *Evidence-Based Midwifery*, 5(2):39–40, 2007.

[222] Singh, R. P., Javaid, M., Haleem, A., Vaishya, R., and Ali, S. Internet of medical things (iomt) for orthopaedic in covid-19 pandemic: Roles, challenges, and applications. *Journal of Clinical Orthopaedics and Trauma*, 11(4):713–717, 2020.

[223] Singh, S. K. and Kumar, P. A comprehensive survey on trajectory schemes for data collection using mobile elements in wsns. *Journal of Ambient Intelligence and Humanized Computing*, 11(1):291–312, 2020.

[224] Smuts, H., Van Der Merwe, A., Loock, M., and Kotzé, P. A framework and methodology for knowledge management system implementation. In *Proceedings of the 2009 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists*, pages 70–79, 2009.

[225] SNaur, . R. B., P. Report on a conference sponsored by the NATO science committee. In *In NATO Software Engineering Conference*, 1968.

[226] Sobeih, A., Chen, W.-P., Hou, J. C., Kung, L.-C., Li, N., Lim, H., Tyan, H.-Y., and Zhang, H. J-sim: A simulation environment for wireless sensor networks. In *38th Annual Simulation Symposium*, pages 175–187. IEEE, 2005.

[227] Sommerville, I. *Ingeniería del software*. Pearson educación, 2005.

[228] Special Mobile Group (SMG). Universal Mobile Telecommunications System (UMTS); Selection procedures for the choice of radio transmission technologies of the UMTS (UMTS 30.03 version 3.2.0) - TR 101 112 V3.2.0 (1998). Technical report, European Telecommunications Standards Institute, 1998.

[229] Staneva, A., Bogossian, F., Pritchard, M., and Wittkowski, A. The effects of maternal depression, anxiety, and perceived stress during pregnancy on preterm birth: A systematic review. *Women and Birth*, 28(3):179–193, 2015.

[230] Sun, L., Li, Y., and Memon, R. A. An open iot framework based on microservices architecture. *China Communications*, 14(2):154–162, 2017.

[231] Swanson, R. A. and Chermack, T. J. *Theory building in applied disciplines*. Berrett-Koehler Publishers, 2013.

[232] SysML. SysML Open Source Project - What is SysML? Who created it? URL https://sysml.org/.

[233] Taherkordi, A., Loiret, F., Rouvoy, R., and Eliassen, F. Optimizing sensor network reprogramming via in situ reconfigurable components. *ACM Transactions on Sensor Networks (TOSN)*, 9(2):1–33, 2013.

[234] Technologies, S. N. Qualnet network simulation software. URL https://www.scalable-networks.com/products/qualnet-network-simulation-software/.

[235] Teng, X.-F., Zhang, Y.-T., Poon, C. C., and Bonato, P. Wearable medical systems for p-health. *IEEE reviews in Biomedical engineering*, 1:62–74, 2008.

[236] Thompson, E. C. G. L. S. A. S. M. . W. D., D. Distributed component object model (dcom). 1997.

[237] Thönes, J. Microservices. *IEEE software*, 32(1):116–116, 2015.

[238] Tichy, W. F. Should computer scientists experiment more? *Computer*, 31(5):32–40, 1998.

[239] TinyOS. Tossim. URL http://tinyos.stanford.edu/tinyos-wiki/index.php/TOSSIM.

[240] Tomovic, S. and Radusinovic, I. Mapping Application Requirements to Virtualization-Enabled Software Defined WSN. *Wireless Personal Communications*, 97(2):1693–1709, 2017.

[241] Tracz, W. Domain-specific software architecture (DSSA) frequently asked questions (FAQ). *ACM SIGSOFT Software Engineering Notes*, 19(2):52–56, 1994.

[242] Unhelkar, B. and Murugesan, S. The enterprise mobile applications development framework. *IT professional*, 12(3):33–39, 2010.

[243] University, R. M. C. Small for gestational age - health encyclopedia - University of Rochester Medical Center. URL https://www.urmc.rochester.edu/encyclopedia/content.aspx?ContentTypeID=90&ContentID=P02411.

[244] Valencia-Flores, M., Resendiz, M., Castãno, V. A., Santiago, V., Campos, R. M., Sandino, S., Valencia, X., Alcocer, J., Garcia Ramos, G., and Bliwise, D. L. Objective and subjective sleep disturbances in patients with systemic lupus erythematosus. *Arthritis & Rheumatism: Official Journal of the American College of Rheumatology*, 42(10): 2189–2193, 1999.

[245] Valtolina, S., Hachem, F., Barricelli, B. R., Belay, E. G., Bonfitto, S., and Mesiti, M.

[246] Van Dam, K., Pitchers, S., and Barnard, M. Body area networks: Towards a wearable future. In *Proc. WWRF kick off meeting, Munich, Germany*, pages 6–7, 2001.

[247] Villegas, N. M., Tamura, G., Müller, H. A., Duchien, L., and Casallas, R. Dynamico: A reference model for governing control objectives and context relevance in self-adaptive software systems. In *Software engineering for self-adaptive systems II*, pages 265–293. Springer, 2013.

[248] Vlissides, J. M. and Linton, M. A. Unidraw: A framework for building domain-specific graphical editors. 8(3), 1990. ISSN 1046-8188.

[249] Vresk, T. and Čavrak, I. Architecture of an interoperable iot platform based on microservices. In *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1196–1201. IEEE, 2016.

[250] Wagh, K. and Thool, R. A comparative study of SOAP vs REST web services provisioning techniques for mobile host. *Journal of Information Engineering and Applications*, 2 (5):12–16, 2012.

[251] Wallace, D. and Hahn, B. H. *Dubois' Lupus Erythematosus and Related Syndromes E-Book: Expert Consult-Online*. Elsevier Health Sciences, 2012.

[252] Wan, J., O'grady, M. J., and O'Hare, G. M. Dynamic sensor event segmentation for real-time activity recognition in a smart home context. *Personal and Ubiquitous Computing*, 19(2):287–301, 2015.

[253] Wang, J., Gao, Y., Liu, W., Sangaiah, A. K., and Kim, H.-J. An intelligent data gathering schema with data fusion supported for mobile sink in wireless sensor networks. *International Journal of Distributed Sensor Networks*, 15(3):1550147719839581, 2019.

[254] Watier, K. A. *Marketing wearable computers to consumers: an examination of early adopter consumers' feelings and attitudes toward wearable computers*. PhD thesis, Citeseer, 2003.

[255] Wei, E. J. and Chan, A. T. Campus: A middleware for automated context-aware adaptation decision making at run time. *Pervasive and Mobile Computing*, 9(1):35–56, 2013.

[256] Weible, C. M. and Sabatier, P. A. *Theories of the policy process*. Routledge, 2018.

[257] Weiser, M. and Brown, J. S. The coming age of calm technology. In *Beyond calculation*, pages 75–85. Springer, 1997.

[258] Weyns, D. Software engineering of self-adaptive systems: an organised tour and future challenges. *Chapter in Handbook of Software Engineering*, 2017.

[259] Weyns, D. and Ahmad, T. Claims and evidence for architecture-based self-adaptation: A systematic literature review. In Drira, K., editor, *Software Architecture*, pages 249–265, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[260] Weyns, D., Malek, S., and Andersson, J. On decentralized self-adaptation: lessons from the trenches and challenges for the future. In *Proceedings of the 2010 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems*, pages 84–93, 2010.

[261] Wink. Wink | help | spotter. URL https://www.wink.com/help/products/quirkyge-spotter-multipurpose-sensor/.

[262] Wooldridge, M. *An introduction to multiagent systems*. John wiley & sons, 2009.

[263] Wrona, K. and Gomez, L. Context-aware security and secure context-awareness in ubiquitous computing environments. *Annales Universitatis Mariae Curie-Sklodowska, sectio AI–Informatica*, 4(1):332–348, 2006.

[264] Wu, H., Hu, J., Sun, J., and Sun, D. Edge computing in an IoT base station system: Reprogramming and real-time tasks. *Complexity*, 2019, 2019.

[265] Wu, X. and Li, F. A method for application reconfiguration in wireless sensor networks. *Transactions of the Institute of Measurement and Control*, 35(3):301–309, 2013.

[266] Yamauchi, M., Jacono, F. J., Fujita, Y., Kumamoto, M., Yoshikawa, M., Campanaro, C. K., Loparo, K. A., Strohl, K. P., and Kimura, H. Effects of environment light during sleep on autonomic functions of heart rate and breathing. *Sleep and Breathing*, 18(4): 829–835, 2014.

[267] Yousefpour, A., Fung, C., Nguyen, T., Kadiyala, K., Jalali, F., Niakanlahiji, A., Kong, J., and Jue, J. P. All one needs to know about fog computing and related edge computing paradigms: A complete survey. *Journal of Systems Architecture*, 98:289–330, 2019.

[268] Yue, W., Hao, W., Liu, P., Liu, T., Ni, M., and Guo, Q. A case—control study on psychological symptoms in sleep apnea-hypopnea syndrome. *The Canadian Journal of Psychiatry*, 48(5):318–323, 2003.

[269] Yue, Y.-G. and He, P. A comprehensive survey on the reliability of mobile wireless sensor networks: Taxonomy, challenges, and future directions. *Information Fusion*, 44: 188–204, 2018.

[270] Zavala, E., Franch, X., and Marco, J. Adaptive monitoring: A systematic mapping. *Information and software technology*, 105:161–189, 2019.

[271] Zhao, D., Ma, H., Tang, S., and Li, X.-Y. COUPON: A cooperative framework for building sensing maps in mobile opportunistic networks. *IEEE transactions on parallel and distributed systems*, 26(2):392–402, 2014.

[272] Zhao, L., Liu, G., Chen, J., and Zhang, Z. Flooding and directed diffusion routing algorithm in wireless sensor networks. In *2009 Ninth International Conference on Hybrid Intelligent Systems*, volume 2, pages 235–239. IEEE, 2009.

[273] Zhou, C. and Krishnamachari, B. Localized topology generation mechanisms for wireless sensor networks. In *GLOBECOM'03. IEEE Global Telecommunications Conference (IEEE Cat. No. 03CH37489)*, volume 3, pages 1269–1273. IEEE, 2003.

[274] Zhu, C., Wu, S., Han, G., Shu, L., and Wu, H. A tree-cluster-based data-gathering algorithm for industrial wsns with a mobile sink. *IEEE Access*, 3:381–396, 2015.

Appendix A

Simulation Results

Table A.1 *System configuration upgrade propagation times for networks consisting of 50 sensor nodes, 1 destination node, and from 4 to 10 intermediary (interm.) nodes. Random Walk mobility model.*

10 interm.		9 interm.		8 interm.		7 interm.		6 interm.		5 interm.		4 interm.	
Sec.	Net. upgraded	Sec.	Net. upgraded	Sec.	Net. upgraded	Sec.	Net. upgraded	Sec.	Net. upgraded	Sec.	Net. upgraded	Sec.	Net. upgraded
2.09	1.67%	2.20	1.69%	2.15	1.72%	2.08	1.75%	2.09	1.79%	2.22	1.82%	2.16	1.85%
2.10	3.33%	2.21	3.39%	2.20	3.45%	2.13	3.51%	2.15	3.57%	2.26	3.64%	2.29	3.70%
2.13	5.00%	2.24	5.08%	2.27	5.17%	2.23	5.26%	2.23	5.36%	2.31	5.45%	2.32	5.56%
2.17	6.67%	2.33	6.78%	2.29	6.90%	2.24	7.02%	2.31	7.14%	2.33	7.27%	2.33	7.41%
2.19	8.33%	2.38	8.47%	2.30	8.62%	2.26	8.77%	2.32	8.93%	2.39	9.09%	2.35	9.26%
2.20	10.00%	2.40	10.17%	2.32	10.34%	2.28	10.53%	2.35	10.71%	2.43	10.91%	2.39	11.11%
2.24	11.67%	2.42	11.86%	2.38	12.07%	2.32	12.28%	2.39	12.50%	2.72	12.73%	2.44	12.96%
2.28	13.33%	2.45	13.56%	2.39	13.79%	2.36	14.04%	2.43	14.29%	2.79	14.55%	2.48	14.81%
2.33	15.00%	2.48	15.25%	2.43	15.52%	2.36	15.79%	2.47	16.07%	2.81	16.36%	2.49	16.67%
2.35	16.67%	2.50	16.95%	2.49	17.24%	2.41	17.54%	2.49	17.86%	2.85	18.18%	2.51	18.52%
2.36	18.33%	2.82	18.64%	2.52	18.97%	2.42	19.30%	2.77	19.64%	2.86	20.00%	2.83	20.37%
2.38	20.00%	3.10	20.34%	2.52	20.69%	2.43	21.05%	2.81	21.43%	3.59	21.82%	3.14	22.22%
2.41	21.67%	3.12	22.03%	2.54	22.41%	2.46	22.81%	3.10	23.21%	3.60	23.64%	3.21	24.07%
2.46	23.33%	3.14	23.73%	3.04	24.14%	2.75	24.56%	3.17	25.00%	3.62	25.45%	3.31	25.93%
2.78	25.00%	3.18	25.42%	3.13	25.86%	2.79	26.32%	3.26	26.79%	3.63	27.27%	3.37	27.78%
2.80	26.67%	3.21	27.12%	3.15	27.59%	2.80	28.07%	3.36	28.57%	3.68	29.09%	3.58	29.63%
2.83	28.33%	3.24	28.81%	3.17	29.31%	2.84	29.82%	3.47	30.36%	3.70	30.91%	3.76	31.48%
2.85	30.00%	3.26	30.51%	3.21	31.03%	2.89	31.58%	4.06	32.14%	3.73	32.73%	4.18	33.33%
3.14	31.67%	3.38	32.20%	3.27	32.76%	2.96	33.33%	4.14	33.93%	3.86	34.55%	4.26	35.19%
3.15	33.33%	3.71	33.90%	3.34	34.48%	3.00	35.09%	4.16	35.71%	4.16	36.36%	4.29	37.04%
3.20	35.00%	3.72	35.59%	3.41	36.21%	3.27	36.84%	4.18	37.50%	4.21	38.18%	4.34	38.89%
3.25	36.67%	3.76	37.29%	3.74	37.93%	3.53	38.60%	4.20	39.29%	4.22	40.00%	4.70	40.74%
3.66	38.33%	4.07	38.98%	3.79	39.66%	3.58	40.35%	4.32	41.07%	4.31	41.82%	4.71	42.59%
3.66	40.00%	4.11	40.68%	4.07	41.38%	3.60	42.11%	4.40	42.86%	4.42	43.64%	4.73	44.44%
3.68	41.67%	4.14	42.37%	4.09	43.10%	3.66	43.86%	4.71	44.64%	4.45	45.45%	4.79	46.30%
3.71	43.33%	4.17	44.07%	4.13	44.83%	3.72	45.61%	4.80	46.43%	4.48	47.27%	4.84	48.15%
3.77	45.00%	4.23	45.76%	4.18	46.55%	3.79	47.37%	4.84	48.21%	4.50	49.09%	5.15	50.00%
3.78	46.67%	4.25	47.46%	4.20	48.28%	3.83	49.12%	4.88	50.00%	4.82	50.91%	5.17	51.85%
3.84	48.33%	4.27	49.15%	4.29	50.00%	4.20	50.88%	4.93	51.79%	5.14	52.73%	5.23	53.70%
4.18	50.00%	4.28	50.85%	4.41	51.72%	4.28	52.63%	4.94	53.57%	5.15	54.55%	5.28	55.56%
4.22	51.67%	4.31	52.54%	4.44	53.45%	4.29	54.39%	4.96	55.36%	5.18	56.36%	5.67	57.41%
4.26	53.33%	4.35	54.24%	4.68	55.17%	4.42	56.14%	5.21	57.14%	5.26	58.18%	5.81	59.26%
4.33	55.00%	4.37	55.93%	4.75	56.90%	4.50	57.89%	5.29	58.93%	5.29	60.00%	6.16	61.11%
4.37	56.67%	4.37	57.63%	4.87	58.62%	4.62	59.65%	5.33	60.71%	5.33	61.82%	6.25	62.96%
4.43	58.33%	4.39	59.32%	4.92	60.34%	4.64	61.40%	5.36	62.50%	5.35	63.64%	6.31	64.81%
4.50	60.00%	4.42	61.02%	5.14	62.07%	4.78	63.16%	5.45	64.29%	5.43	65.45%	6.65	66.67%
4.52	61.67%	4.46	62.71%	5.17	63.79%	4.82	64.91%	5.47	66.07%	5.47	67.27%	7.29	68.52%
4.58	63.33%	4.48	64.41%	5.19	65.52%	5.09	66.67%	5.58	67.86%	5.69	69.09%	8.33	70.37%
4.65	65.00%	4.52	66.10%	5.20	67.24%	5.12	68.42%	5.83	69.64%	5.91	70.91%	10.01	72.22%
4.92	66.67%	4.73	67.80%	5.21	68.97%	5.17	70.18%	5.93	71.43%	6.27	72.73%	11.28	74.07%
5.13	68.33%	4.76	69.49%	5.22	70.69%	5.20	71.93%	5.98	73.21%	6.40	74.55%	11.61	75.93%
5.20	70.00%	4.77	71.19%	5.33	72.41%	5.22	73.68%	6.18	75.00%	6.85	76.36%	11.97	77.78%
5.25	71.67%	4.82	72.88%	5.37	74.14%	5.24	75.44%	6.29	76.79%	7.05	78.18%	12.53	79.63%
5.29	73.33%	5.12	74.58%	5.42	75.86%	5.29	77.19%	6.35	78.57%	8.72	80.00%	13.19	81.48%
5.31	75.00%	5.16	76.27%	5.73	77.59%	5.39	78.95%	6.36	80.36%	9.14	81.82%	13.39	83.33%
5.64	76.67%	5.25	77.97%	5.78	79.31%	5.77	80.70%	6.45	82.14%	9.22	83.64%	14.14	85.19%
5.71	78.33%	5.33	79.66%	5.90	81.03%	5.84	82.46%	6.80	83.93%	9.51	85.45%	15.15	87.04%
5.79	80.00%	5.37	81.36%	5.97	82.76%	6.63	84.21%	7.31	85.71%	10.13	87.27%	16.36	88.89%
5.82	81.67%	5.41	83.05%	6.56	84.48%	7.32	85.96%	8.91	87.50%	11.86	89.09%	17.43	90.74%
5.87	83.33%	5.41	84.75%	6.66	86.21%	9.97	87.72%	11.08	89.29%	11.87	90.91%	19.14	92.59%
6.04	85.00%	5.50	86.44%	6.72	87.93%	10.82	89.47%	11.64	91.07%	12.21	92.73%	19.52	94.44%
7.06	86.67%	5.74	88.14%	6.73	89.66%	12.57	91.23%	13.30	92.86%	12.51	94.55%	24.22	96.30%
7.19	88.33%	6.12	89.83%	8.91	91.38%	12.67	92.98%	13.43	94.64%	12.59	96.36%	64.49	98.15%
8.27	90.00%	8.45	91.53%	8.96	93.10%	12.95	94.74%	14.15	96.43%	17.02	98.18%	73.03	100.00%
8.39	91.67%	8.77	93.22%	9.69	94.83%	13.97	96.49%	16.55	98.21%	17.51	100.00%		
9.03	93.33%	9.35	94.92%	10.06	96.55%	16.23	98.25%	16.69	100.00%				
9.56	95.00%	9.40	96.61%	11.67	98.28%	17.96	100.00%						
10.08	96.67%	13.14	98.31%	14.58	100.00%								
12.10	98.33%	14.04	100.00%										
13.33	100.00%												

Table A.2 *Propagation results for networks composed of 5 fixed source nodes, 10 mobile source nodes, 1 destination node, and from 1 to 6 intermediary nodes. RPGM model.*

Net. upgraded	6 interm. (sec.)	5 interm. (sec.)	4 interm. (sec.)	3 interm. (sec.)	2 interm. (sec.)	1 interm. (sec.)
5%	2.034266	2.1113512	2.15234	2.1246	2.10347582	2.10561823
10%	2.1346234	2.134512	2.162345	2.13623	2.1572759	2.156925123
15%	2.146234	2.26243	2.27234	2.2156232	2.2157295	2.225612385
20%	2.572346	2.42435	3.423466	2.35234	2.30587183	2.31235123
25%	2.6234	2.85234	3.62345	2.36234	2.3158273	2.3517263
30%	3.452344	3.762346	3.7435835	4.1783346	4.85568981	4.605761484
35%	3.623466	3.7772346	3.9958356	4.6344	6.662345	6.42366453
40%	4.624367	4.7234677	5.4345236	6.3534	7.246346	7.23523445
45%	5.234512	5.413466	6.623466	7.47966	8.345667	8.34566345
50%	5.8834	5.672346	7.973457	8.3456	14.34566	15.345666
55%	7.332356	7.634	8.554367	10.24567	14.845688	16.3266418
60%	8.34566	8.62346	10.167832	10.486345	17.935778	18.3564912
65%	9.12667	9.23356	11.724662	17.6434589	19.234666	57.46587612
70%	10.1572455	10.34623	13.572823	17.945	49.52346	
75%	10.52346	11.3452	18.732457	59.565735		
80%	11.67234	12.23467	21.634666			
85%	12.46234	14.62346	21.6634666			
90%	14.52477	15.134678				
95%	15.34662	16.22356				
100%	15.92345	17.34656				

Table A.3 *Propagation results for networks composed of 5 fixed source nodes, 10 mobile source nodes, 1 destination node, and from 1 to 6 intermediary nodes. Manhattan Grid mobility model.*

Net. upgraded	6 interm. (sec.)	5 interm. (sec.)	4 interm. (sec.)	3 interm. (sec.)	2 interm. (sec.)	1 interm. (sec.)
5%	2.31563456	2.4556834	2.563568	2.8176834	2.9056128	3.101926413
10%	2.4663456	2.54	2.635738	2.9723457	3.1056123	3.3056181
15%	2.75673	2.956834	3.15683	3.27243572	3.7056471	3.805764398
20%	2.955637	3.0556834	3.254834	3.6585366	3.9	4.20517283
25%	3.08568345	3.1556834	3.7145683	3.9373546	4.155632	4.905712
30%	3.2172345	3.7556783	3.9435835	4.1783546	4.85568981	5.605761484
35%	3.8173456	3.9356834	4.0958356	4.87835634	5.61765976	5.7056482
40%	4.0157234	4.1168345	4.8758356	5.347347	5.726593	5.860127357
45%	4.1245723	4.8156835	5.2156834	5.6234578	5.9156898	6.470132659
50%	4.7688345	5.3456834	5.857834	7.963467	8.2356898	9.231235
55%	5.21445234	6.35637	6.3458645	9.283456	14.1856849	16.3266418
60%	6.172345	6.415638	8.2345688	10.486345	17.15698401	18.3564912
65%	6.245723	7.4145673	10.25344845	14.6434589	21.35698419	57.46587612
70%	7.1572455	8.2568345	12.58834567	19.273455	48.591783	
75%	7.9724556	9.0966345	16.4568345	59.565735		
80%	8.4723455	10.125683	18.8567834			
85%	8.567235	11.59453	26.94663456			
90%	9.3245672	12.75683456				
95%	10.352345	13.35345634				
100%	11.524562	13.863456				

Table A.4 *Results for data gathering with and without prioritisation. No in-network preprocessing operations are performed on the data. Networks composed of 50 sensor nodes, 1 destination node, and from 1 to 10 intermediary nodes. Random Walk mobility model.*

		Data gathered (KiB)					
		With prioritisation			Without prioritisation		
Intermediary nodes	Data generated in sources for each priority/type (KiB)	Priority 1	Priority 2	Priority 3	Type 1	Type 2	Type 3
1	13007.9245	7144.86269	5306.439713	3813.116972	4771.0769	5291.39388	5161.314635
2	13006.8613	7984.797383	6305.760504	6022.590996	7421.445853	6380.811893	7421.445853
3	13007.7984	9719.326068	8235.147898	7480.909412	8478.461126	7828.064901	8478.461126
4	13007.2684	9837.164856	8418.247443	7862.939146	8900.034951	8836.196393	8900.034951
5	13007.3958	10048.0103	8705.74859	8336.583693	9096.354883	9030.114196	8966.275638
6	13007.5928	10773.44925	9670.949596	9104.85773	9589.593701	9589.593701	9514.017659
7	13008.1728	11191.91418	9695.755708	9215.334033	10294.49313	9832.58173	9572.42324
8	13007.8273	11315.3854	10000.29724	9352.775763	10743.13645	10294.49313	10500.14842
9	13007.1195	11322.37065	10177.58224	10000.29724	11150.4796	10500.08338	10564.52897
10	13007.4483	12750.95295	10432.61561	10070.96929	11605.16293	10824.68746	10760.24187

Table A.5 *Data gathered when in-network preprocessing is applied vs. when it is not applied. Networks composed of 50 sensor nodes, 1 destination node, and from 1 to 10 intermediary nodes. Random Walk mobility model.*

		Data gathered (KiB)	
Intermediary nodes	Total data generated (KiB)	Without in-network preprocessing	With in-network preprocessing
1	39023.7735	16264.41937	20449.50741
2	39020.5839	20311.48859	24139.30817
3	39023.3952	25435.1368	29466.10356
4	39021.8052	26117.03407	31378.44974
5	39022.1874	27089.24152	33110.06929
6	39022.7784	29548.50307	34865.73137
7	39024.5184	30103.57853	36013.05533
8	39023.4819	30668.22923	36860.90217
9	39021.3585	31498.30072	37294.70401
10	39022.3449	33253.32045	38214.77377

Table A.6 *Data gathering under RPGM model. Networks composed of 10 mobile source nodes, 5 fixed source nodes, 1 destination node, and from 0 to 6 intermediary nodes. Without in-network preprocessing operations.*

Intermediary nodes	Data generated in fixed sources (KiB)	Remaining data in fixed sources (KiB)	Data generated in mobile sources (KiB)	Remaining data in mobile sources (KiB)	Data received in destinations (KiB)	Remaining data in intermediaries (KiB)
0	2328.548311	2241.390107	6238.158809	2296.542598	4029.56543	0
1	2327.354316	1562.681465	6239.553496	500.6960742	3206.157227	3298.789063
2	2326.756738	1341.712793	6238.105576	201.4454199	3115.087891	3908.369141
3	2327.136006	1209.499287	6238.468662	121.5790137	3365.405273	3873.261719
4	2326.828047	1138.141523	6239.76833	71.03786133	3675.605469	3685.004883
5	2328.387285	1082.444902	6238.029424	67.32629883	3993.47168	3429.448242
6	2327.800264	1067.683076	6238.312383	50.68054688	4010.175781	3445.976563

Table A.7 *Data gathering under RPGM model. Networks composed of 10 mobile source nodes, 5 fixed source nodes, 2 destination nodes, and from 0 to 6 intermediary nodes. Without in-network preprocessing operations.*

Intermediary nodes	Data generated in fixed sources (KiB)	Remaining data in fixed sources (KiB)	Data generated in mobile sources (KiB)	Remaining data in mobile sources (KiB)	Data received in destinations (KiB)	Remaining data in intermediaries (KiB)
0	2327.980596	2100.338994	6238.165469	1078.292422	5388.330078	0
1	2327.720801	1487.730566	6239.193584	294.8283496	4427.910156	2357.895508
2	2327.129668	1297.442168	6238.269561	153.6894824	4508.466797	2607.729492
3	2327.804824	1181.276504	6238.227295	88.26147461	4844.204102	2455.351563
4	2327.867158	1109.893525	6238.882227	62.93984375	5182.255859	2216.005859
5	2327.289961	1056.249922	6239.193848	52.72412109	5477.158203	1983.945313
6	2329.020166	1049.967432	6239.151279	34.04385742	5435.239258	2053.149414

Table A.8 *Data gathering under RPGM model. Networks composed of 10 mobile source nodes, 5 fixed source nodes, 3 destination nodes, and from 0 to 6 intermediary nodes. Without in-network preprocessing operations.*

Intermediary nodes	Data generated in fixed sources (KiB)	Remaining data in fixed sources (KiB)	Data generated in mobile sources (KiB)	Remaining data in mobile sources (KiB)	Data received in destinations (KiB)	Remaining data in intermediaries (KiB)
0	2328.035371	2016.414277	6239.51627	576.3815039	5975.74707	0
1	2328.05334	1439.967402	6237.591836	207.9189844	5547.744141	1371.660156
2	2327.946279	1250.602529	6239.142461	112.1746875	5651.777344	1554.643555
3	2328.03874	1125.978193	6238.731377	72.15911133	5966.850586	1404.360352
4	2327.732598	1060.398613	6238.844043	44.24443359	6293.618164	1171.645508
5	2327.508984	1010.01875	6239.081533	41.4252832	6502.919922	1016.601563
6	2327.783984	993.8728516	6239.03373	29.65873047	6510.463867	1038.095703

Table A.9 *Data gathering under RPGM model. Networks composed of 10 mobile source nodes, 5 fixed source nodes, 4 destination nodes, and from 0 to 6 intermediary nodes. Without in-network preprocessing operations.*

Intermediary nodes	Data generated in fixed sources (KiB)	Remaining data in fixed sources (KiB)	Data generated in mobile sources (KiB)	Remaining data in mobile sources (KiB)	Data received in destinations (KiB)	Remaining data in intermediaries (KiB)
0	2328.192734	1750.658555	6239.081719	574.6090625	6242.988281	0
1	2327.678213	1242.536611	6238.339531	199.335625	5665.004883	1460.81543
2	2328.52002	1092.943848	6239.444541	99.95723633	5777.451172	1599.716797
3	2327.703398	984.3733203	6238.122568	82.15577148	6158.374023	1344.160156
4	2327.96041	932.1986914	6239.748408	50.01696289	6454.482422	1135.805664
5	2327.467617	896.9891016	6238.697822	41.2124707	6635.898438	996.0302734
6	2327.581133	878.3330859	6238.719551	28.23126953	6715.092773	949.5117188

Table A.10 *Data gathering under RPGM model. Networks composed of 10 mobile source nodes, 5 fixed source nodes, 5 destination nodes, and from 0 to 6 intermediary nodes. Without in-network preprocessing operations.*

Intermediary nodes	Data generated in fixed sources (KiB)	Remaining data in fixed sources (KiB)	Data generated in mobile sources (KiB)	Remaining data in mobile sources (KiB)	Data received in destinations (KiB)	Remaining data in intermediaries (KiB)
0	2327.059854	1586.635049	6239.524609	294.8957031	6686.5625	0
1	2327.691445	1159.141641	6237.999004	130.2304492	6685.678711	592.5732422
2	2328.21626	1010.140088	6240.397969	78.14699219	6849.760742	632.8808594
3	2327.075879	901.5631836	6238.823145	64.80458984	7035.478516	566.8261719
4	2328.148535	859.349707	6239.539521	43.04538086	7209.785156	459.5947266
5	2327.617852	823.9459766	6240.202451	38.15166992	7337.96875	372.0654297
6	2327.615898	813.53875	6239.714307	26.23774414	7328.974609	403.3642578

Table A.11 *Data gathering under RPGM model. Networks composed of 10 mobile source nodes, 5 fixed source nodes, 6 destination nodes, and from 0 to 6 intermediary nodes. Without in-network preprocessing operations.*

Intermediary nodes	Data generated in fixed sources (KiB)	Remaining data in fixed sources (KiB)	Data generated in mobile sources (KiB)	Remaining data in mobile sources (KiB)	Data received in destinations (KiB)	Remaining data in intermediaries (KiB)
0	2328.07543	1398.00707	6239.81307	242.864824	6928.41309	0
1	2327.2161	1006.53251	6238.55314	109.944746	6979.00879	472.158203
2	2327.39004	860.563867	6239.89527	76.1062109	7127.96875	505.136719
3	2326.83589	771.845654	6239.19407	62.1725879	7293.91113	441.113281
4	2327.0347	728.343291	6239.11604	42.9099805	7444.28711	354.248047
5	2327.33979	704.165967	6238.77646	35.3438379	7537.92969	293.120117
6	2326.99253	712.847021	6240.32855	23.5414453	7559.59961	276.450195

Table A.12 *Data gathering under Manhattan Grid mobility model. Networks composed of 10 mobile source nodes, 5 fixed source nodes, 1 destination node, and from 0 to 6 intermediary nodes. Without in-network preprocessing operations.*

Intermediary nodes	Data generated in fixed sources (KiB)	Remaining data in fixed sources (KiB)	Data generated in mobile sources (KiB)	Remaining data in mobile sources (KiB)	Data received in destinations (KiB)	Remaining data in intermediaries (KiB)
0	2327.47959	2240.44346	6238.03551	3985.71617	2339.35547	0
1	2327.45541	1561.79623	6239.20123	2605.86627	2377.19727	2021.79688
2	2326.96231	1113.96915	6239.66515	1692.45323	2393.97949	3366.22559
3	2327.95512	813.858438	6238.07063	1078.84212	2436.0498	4237.27539
4	2328.10246	631.037031	6238.75132	711.20249	2486.17676	4738.4375
5	2327.55723	485.486914	6238.17241	487.889209	2626.19629	4966.15723
6	2327.62121	408.392695	6239.28438	333.844922	2742.61719	5082.05078

Table A.13 *Data gathering under Manhattan Grid mobility model. Networks composed of 10 mobile source nodes, 5 fixed source nodes, 2 destination nodes, and from 0 to 6 intermediary nodes. Without in-network preprocessing operations.*

Intermediary nodes	Data generated in fixed sources (KiB)	Remaining data in fixed sources (KiB)	Data generated in mobile sources (KiB)	Remaining data in mobile sources (KiB)	Data received in destinations (KiB)	Remaining data in intermediaries (KiB)
0	2328.54639	2100.93896	6238.9824	2390.62791	4075.96191	0
1	2327.53375	1476.7818	6238.77811	1526.35623	4143.2373	1419.93652
2	2328.04162	1061.2301	6239.12009	985.570283	4226.03516	2294.32617
3	2327.95813	790.712031	6239.22581	649.299053	4355.78613	2771.38672
4	2328.48434	614.875938	6238.85325	441.978252	4602.77344	2907.70996
5	2327.95634	474.753213	6238.22788	311.308936	4835.87402	2944.24805
6	2328.25329	401.241572	6239.83432	237.197598	5058.14941	2871.49902

Table A.14 *Data gathering under Manhattan Grid mobility model. Networks composed of 10 mobile source nodes, 5 fixed source nodes, 3 destination nodes, and from 0 to 6 intermediary nodes. Without in-network preprocessing operations.*

Intermediary nodes	Data generated in fixed sources (KiB)	Remaining data in fixed sources (KiB)	Data generated in mobile sources (KiB)	Remaining data in mobile sources (KiB)	Data received in destinations (KiB)	Remaining data in intermediaries (KiB)
0	2327.49161	2015.78263	6239.09604	1348.62729	5202.17773	0
1	2328.0191	1437.41363	6239.44867	848.05707	5382.04102	899.956055
2	2328.07443	1037.13693	6237.89233	575.36792	5530.40039	1423.06152
3	2327.33787	766.839824	6238.93029	392.060176	5777.49023	1629.87793
4	2326.95029	597.443457	6239.09768	289.55666	6017.6416	1661.40625
5	2328.19035	463.620039	6239.70171	227.050342	6267.85156	1609.37012
6	2328.27814	388.214668	6238.43298	177.212275	6450.37598	1550.9082

Table A.15 *Data gathering under Manhattan Grid mobility model. Networks composed of 10 mobile source nodes, 5 fixed source nodes, 4 destination nodes, and from 0 to 6 intermediary nodes. Without in-network preprocessing operations.*

Intermediary nodes	Data generated in fixed sources (KiB)	Remaining data in fixed sources (KiB)	Data generated in mobile sources (KiB)	Remaining data in mobile sources (KiB)	Data received in destinations (KiB)	Remaining data in intermediaries (KiB)
0	2327.9788	1750.33231	6238.59111	994.294238	5821.94336	0
1	2327.38753	1243.82308	6239.95617	632.846797	6068.67188	622.001953
2	2327.68561	912.45123	6238.7665	436.55459	6325.93262	891.513672
3	2327.8272	698.774463	6239.12663	313.408857	6590	964.770508
4	2327.94055	546.763789	6240.62675	227.428506	6854.43848	939.936523
5	2327.99484	433.614961	6238.62102	176.443281	7064.36035	892.197266
6	2327.95611	362.980527	6239.265	138.791367	7251.58203	813.867188

Table A.16 *Data gathering under Manhattan Grid mobility model. Networks composed of 10 mobile source nodes, 5 fixed source nodes, 5 destination nodes, and from 0 to 6 intermediary nodes. Without in-network preprocessing operations.*

Intermediary nodes	Data generated in fixed sources (KiB)	Remaining data in fixed sources (KiB)	Data generated in mobile sources (KiB)	Remaining data in mobile sources (KiB)	Data received in destinations (KiB)	Remaining data in intermediaries (KiB)
0	2327.41604	1588.77346	6239.37462	551.547471	6426.46973	0
1	2327.77881	1126.4751	6238.86728	369.467861	6791.27441	279.428711
2	2328.17104	822.526514	6238.91661	265.606064	7099.53125	379.423828
3	2327.0699	633.398027	6238.28023	199.88668	7315.84961	416.21582
4	2327.70385	507.30834	6238.72937	152.425654	7498.76465	407.93457
5	2327.44361	404.811777	6239.52601	125.33167	7652.84668	383.979492
6	2328.52836	346.301797	6238.5108	109.096738	7759.19922	352.441406

Table A.17 *Data gathering under Manhattan Grid mobility model. Networks composed of 10 mobile source nodes, 5 fixed source nodes, 6 destination nodes, and from 0 to 6 intermediary nodes. Without in-network preprocessing operations.*

Intermediary nodes	Data generated in fixed sources (KiB)	Remaining data in fixed sources (KiB)	Data generated in mobile sources (KiB)	Remaining data in mobile sources (KiB)	Data received in destinations (KiB)	Remaining data in intermediaries (KiB)
0	2327.16811	1396.85072	6238.95518	335.205176	6834.06738	0
1	2327.19176	998.217148	6239.12853	245.036729	7202.0459	121.020508
2	2327.3857	737.664023	6238.77448	188.808662	7464.80957	174.87793
3	2327.85676	573.335273	6237.78208	149.945166	7646.5918	195.766602
4	2327.80894	460.284521	6238.13679	119.142646	7786.93848	199.580078
5	2327.22753	368.472646	6239.21936	105.151973	7892.70996	200.112305
6	2327.84764	313.472637	6240.80557	92.5926758	7982.7002	179.887695

Table A.18 *Data gathering under RPGM model. Networks composed of 10 mobile source nodes, 5 fixed source nodes, 1 destination node, and from 0 to 6 intermediary nodes. With in-network preprocessing operations.*

Intermediary nodes	Preprocessed data to be gathered (KiB)	Remaining data in fixed sources (KiB)	Remaining data in mobile sources (KiB)	Data received in destinations (KiB)	Remaining data in intermediaries (KiB)
0	856.670712	315.5076631	229.6542598	311.5087891	0
1	856.6907812	156.1265448	50.06960742	468.7338722	181.7607568
2	856.4862314	156.8563379	20.14454199	493.4026171	186.0827344
3	856.5604668	132.2443622	12.15790137	524.1272266	188.0309766
4	856.6596377	81.70819527	7.103786133	565.1723877	202.6752686
5	856.6416709	61.56529102	6.732629883	563.6945703	224.6491797
6	856.6112647	57.74975291	5.068054688	567.1080469	226.6854102

Table A.19 *Data gathering under RPGM model. Networks composed of 10 mobile source nodes, 5 fixed source nodes, 2 destination nodes, and from 0 to 6 intermediary nodes. With in-network preprocessing operations.*

Intermediary nodes	Preprocessed data to be gathered (KiB)	Remaining data in fixed sources (KiB)	Remaining data in mobile sources (KiB)	Data received in destinations (KiB)	Remaining data in intermediaries (KiB)
0	856.6146065	305.9943487	107.8292422	442.7910156	0
1	856.6914385	177.9673925	29.48283496	544.0921094	105.1491016
2	856.5399229	151.4356231	15.36894824	578.9350586	110.800293
3	856.6032119	107.9508926	8.826147461	620.7435058	119.082666
4	856.6749385	75.75839552	6.293984375	654.3698877	120.2526709
5	856.6483809	62.31688669	5.272412109	663.8361524	125.2229297
6	856.8171445	44.92398926	3.404385742	659.8481885	148.640581

Table A.20 *Data gathering under RPGM model. Networks composed of 10 mobile source nodes, 5 fixed source nodes, 3 destination nodes, and from 0 to 6 intermediary nodes. With in-network preprocessing operations.*

Intermediary nodes	Preprocessed data to be gathered (KiB)	Remaining data in fixed sources (KiB)	Remaining data in mobile sources (KiB)	Data received in destinations (KiB)	Remaining data in intermediaries (KiB)
0	856.7551641	244.3425996	57.63815039	554.7744141	0
1	856.5645176	168.9347285	20.79189844	614.991211	51.84667971
2	856.708874	144.9967764	11.21746875	644.4374609	56.05716796
3	856.6770117	134.7218428	7.215911133	656.1569824	58.5822754
4	856.6576641	85.70538874	4.424443359	691.0865234	75.44130858
5	856.6590517	61.78849598	4.14252832	710.6794873	80.04854006
6	856.6817714	47.20515615	2.965873047	719.4507031	87.06003908

Table A.21 *Data gathering under RPGM model. Networks composed of 10 mobile source nodes, 5 fixed source nodes, 4 destination nodes, and from 0 to 6 intermediary nodes. With in-network preprocessing operations.*

Intermediary nodes	Preprocessed data to be gathered (KiB)	Remaining data in fixed sources (KiB)	Remaining data in mobile sources (KiB)	Data received in destinations (KiB)	Remaining data in intermediaries (KiB)
0	856.7274453	232.7660508	57.46090625	566.5004883	0
1	856.6017744	163.9719228	19.9335625	616.9185938	55.77769531
2	856.7964561	131.3603028	9.995723633	658.4697265	56.97070313
3	856.5825966	110.487625	8.215577148	679.9533056	57.92608886
4	856.7708818	71.90492771	5.001696289	702.44104	77.42321779
5	856.6165439	42.82391003	4.12124707	731.7100978	77.96128905
6	856.6300684	22.32598445	2.823126953	749.8954004	81.58555665

Table A.22 *Data gathering under RPGM model. Networks composed of 10 mobile source nodes, 5 fixed source nodes, 5 destination nodes, and from 0 to 6 intermediary nodes. With in-network preprocessing operations.*

Intermediary nodes	Preprocessed data to be gathered (KiB)	Remaining data in fixed sources (KiB)	Remaining data in mobile sources (KiB)	Data received in destinations (KiB)	Remaining data in intermediaries (KiB)
0	856.6584463	158.6010049	29.48957031	668.5678711	0
1	856.5690449	137.683207	13.02304492	683.5388672	22.32392578
2	856.8614229	123.7342237	7.814699219	703.5308301	21.78166992
3	856.5899024	100.6021192	6.480458984	725.1488038	24.35852051
4	856.7688056	74.80313472	4.304538086	745.9188672	31.74226563
5	856.7820303	60.81207819	3.815166992	757.7855371	34.36924805
6	856.7330205	57.02428515	2.623774414	761.6436328	35.44132813

Table A.23 *Data gathering under RPGM model. Networks composed of 10 mobile source nodes, 5 fixed source nodes, 6 destination nodes, and from 0 to 6 intermediary nodes. With in-network preprocessing operations.*

Intermediary nodes	Preprocessed data to be gathered (KiB)	Remaining data in fixed sources (KiB)	Remaining data in mobile sources (KiB)	Data received in destinations (KiB)	Remaining data in intermediaries (KiB)
0	856.78885	139.6610586	24.2864824	692.841309	0
1	856.576924	120.0365509	10.9944746	709.7882374	15.75766112
2	856.728531	107.0090232	7.61062109	726.2804004	15.82848632
3	856.602996	85.56981951	6.21725879	744.269531	20.54638673
4	856.615074	63.78403685	4.29099805	762.9554688	25.5845703
5	856.611625	52.06845191	3.53438379	775.2468166	25.76197267
6	856.732108	47.90433057	2.35414453	779.0884571	27.38517577

Table A.24 *Data gathering under Manhattan Grid mobility model. Networks composed of 10 mobile source nodes, 5 fixed source nodes, 1 destination node, and from 0 to 6 intermediary nodes. With in-network preprocessing operations.*

Intermediary nodes	Preprocessed data to be gathered (KiB)	Remaining data in fixed sources (KiB)	Remaining data in mobile sources (KiB)	Data received in destinations (KiB)	Remaining data in intermediaries (KiB)
0	856.55151	224.044346	398.571617	233.935547	0
1	856.665664	156.179623	260.586627	334.7659772	105.1334378
2	856.662746	111.396915	169.245323	407.7092285	168.3112795
3	856.602575	81.3858438	107.884212	451.2314741	216.1010449
4	856.685378	63.1037031	71.120249	474.1535645	248.3078615
5	856.572964	48.5486914	48.7889209	484.405977	274.829375
6	856.690559	40.8392695	33.3844922	497.5437502	284.9230468

Table A.25 *Data gathering under Manhattan Grid mobility model. Networks composed of 10 mobile source nodes, 5 fixed source nodes, 2 destination nodes, and from 0 to 6 intermediary nodes. With in-network preprocessing operations.*

Intermediary nodes	Preprocessed data to be gathered (KiB)	Remaining data in fixed sources (KiB)	Remaining data in mobile sources (KiB)	Data received in destinations (KiB)	Remaining data in intermediaries (KiB)
0	856.752879	210.093896	239.062791	407.596191	0
1	856.631186	147.67818	152.635623	483.9006195	72.41676252
2	856.716171	106.12301	98.5570283	537.3198245	114.7163085
3	856.718394	79.0712031	64.9299053	566.270835	146.44645
4	856.733759	61.4875938	44.1978252	599.8474221	151.2009179
5	856.618422	47.4753213	31.1308936	616.0785642	161.9336428
6	856.808761	40.1241572	23.7197598	629.4530265	163.5118165

Table A.26 *Data gathering under Manhattan Grid mobility model. Networks composed of 10 mobile source nodes, 5 fixed source nodes, 3 destination nodes, and from 0 to 6 intermediary nodes. With in-network preprocessing operations.*

Intermediary nodes	Preprocessed data to be gathered (KiB)	Remaining data in fixed sources (KiB)	Remaining data in mobile sources (KiB)	Data received in destinations (KiB)	Remaining data in intermediaries (KiB)
0	856.658765	201.578263	134.862729	520.217773	0
1	856.746777	143.741363	84.805707	574.2023442	53.9973633
2	856.596676	103.713693	57.536792	617.800781	77.54541
3	856.626816	76.6839824	39.2060176	662.4684324	78.2683836
4	856.604797	59.7443457	28.955666	682.6081686	85.29661636
5	856.789206	46.3620039	22.7050342	696.3448243	91.37734375
6	856.671112	38.8214668	17.7212275	705.5954981	94.53291994

Table A.27 *Data gathering under Manhattan Grid mobility model. Networks composed of 10 mobile source nodes, 5 fixed source nodes, 4 destination nodes, and from 0 to 6 intermediary nodes. With in-network preprocessing operations.*

Intermediary nodes	Preprocessed data to be gathered (KiB)	Remaining data in fixed sources (KiB)	Remaining data in mobile sources (KiB)	Data received in destinations (KiB)	Remaining data in intermediaries (KiB)
0	856.656991	175.033231	99.4294238	582.194336	0
1	856.73437	124.382308	63.2846797	634.8572759	34.21010742
2	856.645211	91.245123	43.655459	676.1680667	45.57656253
3	856.695383	69.8774463	31.3408857	706.4062012	49.07084963
4	856.85673	54.6763789	22.7428506	729.6208646	49.81663572
5	856.661586	43.3614961	17.6443281	744.5229247	51.13283692
6	856.722111	36.2980527	13.8791367	753.9456152	52.59930665

Table A.28 *Data gathering under Manhattan Grid mobility model. Networks composed of 10 mobile source nodes, 5 fixed source nodes, 5 destination nodes, and from 0 to 6 intermediary nodes. With in-network preprocessing operations.*

Intermediary nodes	Preprocessed data to be gathered (KiB)	Remaining data in fixed sources (KiB)	Remaining data in mobile sources (KiB)	Data received in destinations (KiB)	Remaining data in intermediaries (KiB)
0	856.679066	158.877346	55.1547471	642.646973	0
1	856.664609	112.64751	36.9467861	691.701733	15.36857911
2	856.708765	82.2526514	26.5606064	728.8636719	19.03183592
3	856.535013	63.3398027	19.988668	753.6235889	19.58295409
4	856.643322	50.730834	15.2425654	769.4429152	21.22700682
5	856.696962	40.4811777	12.533167	781.2966114	22.38600585
6	856.703916	34.6301797	10.9096738	787.5038575	23.66020506

Table A.29 *Data gathering under Manhattan Grid mobility model. Networks composed of 10 mobile source nodes, 5 fixed source nodes, 6 destination nodes, and from 0 to 6 intermediary nodes. With in-network preprocessing operations.*

Intermediary nodes	Preprocessed data to be gathered (KiB)	Remaining data in fixed sources (KiB)	Remaining data in mobile sources (KiB)	Data received in destinations (KiB)	Remaining data in intermediaries (KiB)
0	856.612329	139.685072	33.5205176	683.406738	0
1	856.632029	99.8217148	24.5036729	726.1345949	6.172045908
2	856.616018	73.7664023	18.8808662	754.2548145	9.71393553
3	856.563884	57.3335273	14.9945166	774.4426761	9.79316408
4	856.594573	46.0284521	11.9142646	788.6728519	9.9790039
5	856.644689	36.8472646	10.5151973	799.1023632	10.1798633
6	856.865321	31.3472637	9.25926758	805.0525004	11.20628908

Table A.30 *Average battery level of network nodes over time when in-network preprocessing is applied vs. when it is not applied.*

Sec.	With in-network preprocessing (mAh)	Without in-network preprocessing (mAh)
0	1.92	1.92
500	1.84253074	1.651131389
1000	1.720187469	1.443092927
1500	1.592994385	1.265054435
2000	1.489941996	1.107505726
2500	1.451565008	1.004022743
3000	1.326721014	0.9064349185
3500	1.288283324	0.8514849593
4000	1.258187336	0.8104174433
4500	1.148074298	0.7616005105
5000	1.09673915	0.7616005105
5500	1.09673915	0.7274138642
6000	1.042557627	0.7274138642
6500	1.015847544	0.7274138642
7000	0.9642481509	0.7274138642
7500	0.9563206253	0.6980448577
8000	0.9410087111	0.6222571577
8500	0.89202666	0.5712924469
9000	0.89202666	0.5173125243
9500	0.89202666	0.4078303812
10000	0.89202666	0.2356733122
10500	0.8676957047	0.06971999423
11000	0.8676957047	
11500	0.8120788209	
12000	0.7951782803	
12500	0.7951782803	
13000	0.7880064723	
13500	0.7816971951	
14000	0.7121508955	
14500	0.7121508955	
15000	0.6463077127	
15500	0.6463077127	
16000	0.6016750248	
16500	0.5291758691	
17000	0.4852687963	
17500	0.4521342129	
18000	0.3371213896	
18500	0.2784574608	
19000	0.2181538507	
19500	0.07204811897	

UNIVERSIDAD
DE GRANADA